

# 网易微专业之《前端开发工程师》

## 学习笔记

开始时间：2016.03.01

## 《DOM 编程艺术》

### 基础篇（三）

#### 多媒体和图形编程（音频与视频）



#### 基本用法

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

```
<video src=" movie.mov" width=320 height=240> </video>
```

#### **<audio>**兼容用法

```
<audio>
```

```
<source src=" music.mp3" type=" audio/mpeg" >
```

```
<source src=" music.wav" type=" audio/x-wav" >
```

```
<source src=" music.ogg" type=" audio/ogg" >
```

本笔记由西风潇潇编写，欢迎浏览博客访问更多内容：<http://www.xifengxx.com>

</audio>

## <video> 兼容用法

<video>

<source src="movie.webm" type="video/webm; codecs='vp8,vorbis' ">

<source src="movie.mp4" type="video/mp4;

codecs='avc1.42E01E,mp4a.40.2' ">

</video>

## 多媒体格式支持类型

音频：

[https://en.wikipedia.org/wiki/HTML5\\_Audio#Supported\\_audio\\_coding\\_formats](https://en.wikipedia.org/wiki/HTML5_Audio#Supported_audio_coding_formats)

视频：

[https://en.wikipedia.org/wiki/HTML5\\_video#Browser\\_support](https://en.wikipedia.org/wiki/HTML5_video#Browser_support)

## 多媒体格式兼容性

对于 Video 而言，需先在页面中插入 video 标签，通过 JS 去获取 Video 标签的 DOM 对象，再通过调用 DOM 对象的 canPlayType 方法去检测。

对于 Audio 而言，可以直接用 Audio 构造函数生成一个对象来检测。



## Audio/video 的 HTML 属性

<audio>和<video>除了界面上显示不一样，他们大部分的属性和方法是一样的。

属性	是否必须	默认值	备注
src	是		音频文件的URL
controls	否	false	向用户显示控件。
autoplay	否	false	音频在就绪后马上播放
preload	否	none	可取值为“none”、“metadata”、“auto”。音频在页面加载时进行加载，并预备播放。如果使用“autoplay”，则忽略该属性。
loop	否	false	每当音频结束时重新开始播放。

- Preload:none-不进行预加载
- Preload:metadata-进行预加载，但只加载媒体元信息
- Preload:auto-预加载资源信息（全部信息）

## 控制多媒体播放

### 方法

- load() 加载媒体内容
- play() 开始播放
- pause() 暂停播放

### 属性

- playbackRate 播放速度，0-1 为慢速播放，1 为正常速度播放，大于 1 为快速播放。
- currentTime 播放进度，以秒为单位。
- volume 音量，取值范围 0 到 1
- muted 静音（布尔值）
- paused 暂停（布尔值）
- seeking 跳转（布尔值）
- ended 播放完成（布尔值）
- duration 媒体时长（数值）
- initialTime 媒体开始时间

### 事件

- loadstart 开始请求媒体内容
- loadmetadata 媒体元数据已经加载完成（时长，编码格式等）

本笔记由西风潇潇编写，欢迎浏览博客访问更多内容：<http://www.xifengxx.com>

- `canplay` 加载一些内容，可以开始播放
- `play` 调用 `play()` 或设置 `autoplay`
- `waiting` 缓冲数据不够，暂停播放
- `playing` 正在播放
- 相关事件列表：

[https://www.w3.org/wiki/HTML/Elements/audio#Media\\_Events](https://www.w3.org/wiki/HTML/Elements/audio#Media_Events)

## Web Audio API

W3C 官网定义

<http://webaudio.github.io/web-audio-api/>

Mozilla 官方音频教程

[https://developer.mozilla.org/en-US/docs/Web/API/Web\\_Audio\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API)

第三方教程

<http://www.html5rocks.com/en/tutorials/webaudio/intro/>

<http://webaudioapi.com/>

## canvas

### 基本用法

```
<canvas id="tutorial" width="300" height="150"></canvas>
```

画布 `<canvas>` 的默认宽高为 300 与 150，但是同样可以使用 CSS 设定宽高。但因为 CSS 与 JavaScript 在渲染工程中有速度的差异，所以**不建议使用** CSS 对 `<canvas>` 的宽高做设定。

### 渲染上下文对象

canvas 起初是空白的。为了展示，首先脚本需要找到渲染上下文，然后在它的上面绘制。`<canvas>` 元素有一个做 `getContext()` 的方法，这个方法是用来获得渲染上下文和它的绘画功能。`getContext()` 只有一个参数，上下文的格式。

下面代码中的 `ctx` 即为渲染上下文对象。

```
var canvas = document.getElementById('tutorial');

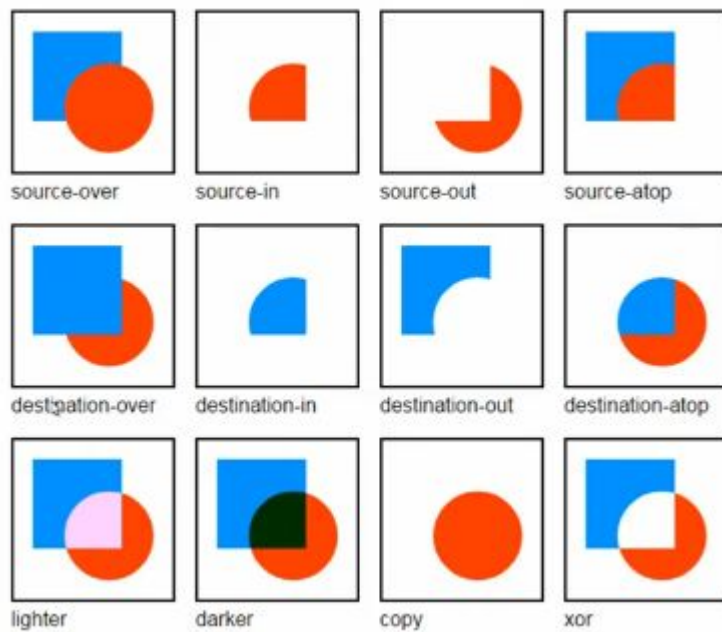
var ctx = canvas.getContext('2d');

// 绘画 canvas 的属性

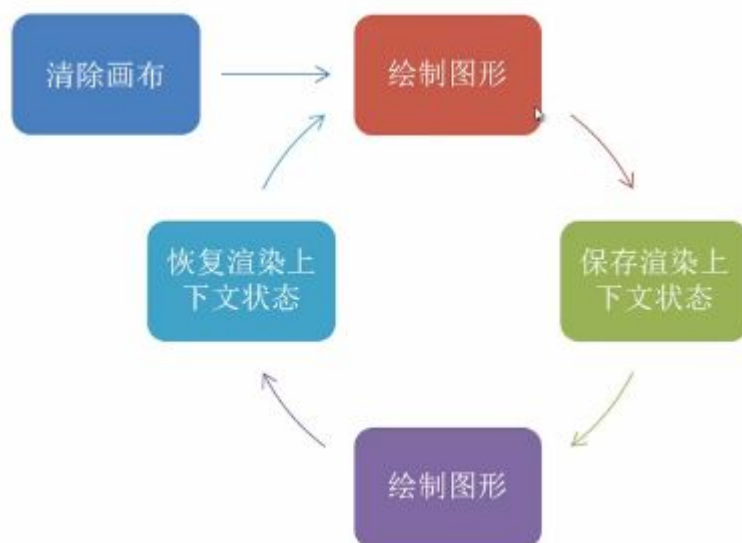
ctx.globalCompositeOperation = 'destination-over';
```

`globalCompositeOperation` (全局的组合操作) 为对于 `canvas` 绘画时的渲染属性

设置，具体表现如下图：



## 基本绘图的步骤



本笔记由西风潇潇编写，欢迎浏览博客访问更多内容：<http://www.xifengxx.com>

上图的实现代码如下：

```
var sun = new Image();
var moon = new Image();
var earth = new Image();
function init(){
    sun.src = 'Canvas_sun.png';
    moon.src = 'Canvas_moon.png';
    earth.src = 'Canvas_earth.png';
    window.requestAnimationFrame(draw);
}

function draw() {
}

init();
```

```
var ctx = document.getElementById('canvas').getContext('2d');

ctx.globalCompositeOperation = 'destination-over';
ctx.clearRect(0,0,300,300); // clear canvas

ctx.fillStyle = 'rgba(0,0,0,0.4)';
ctx.strokeStyle = 'rgba(0,153,255,0.4)';
ctx.save();
ctx.translate(150,150);

// Earth
var time = new Date();
ctx.rotate( ((2*Math.PI)/60)*time.getSeconds() + ((2*Math.PI)/60000)*time.getMilliseconds() );
ctx.translate(105,0);
ctx.fillRect(0,-12,50,24); // Shadow
ctx.drawImage(earth,-12,-12);

// Moon
ctx.rotate( ((2*Math.PI)/6)*time.getSeconds() + ((2*Math.PI)/6000)*time.getMilliseconds() );
ctx.translate(0,28.5);
ctx.drawImage(moon,-3.5,-3.5);

ctx.restore();

ctx.beginPath();
ctx.arc(150,150,105,0,Math.PI*2,false); // Earth orbit
ctx.stroke();

ctx.drawImage(sun,0,0,300,300);

window.requestAnimationFrame(draw);
```

清空画布

保存画布状态

绘制图形

恢复画布状态

## Canvas 完整教程：

### Mozilla 官方教程：

[https://developer.mozilla.org/zh-CN/docs/Web/API/Canvas\\_API/Tutorial](https://developer.mozilla.org/zh-CN/docs/Web/API/Canvas_API/Tutorial)

### canvas 引擎

目前市面上有哪些比较有名的基于 canvas 的引擎，他们各有什么特点，合适做什么应用？

- Laro: 是一个基于 html5 canvas 的用于平面 2d 或者 2.5d 游戏制作的轻量级游戏引擎。
- X-Canvas: 一款跨平台的 HTML5 游戏引擎，提供手机游戏开发的完整解决方案。包含了加速引擎，游戏框架，物理引擎

- CutJS: 一款专门用于跨平台游戏开发的开源 2D HTML5 渲染引擎，轻量级、快速、可交互
- cocos2djs: 一个游戏框架，api 广泛
- Phaser: 一款专门用于桌面及移动 HTML5 2D 游戏开发的开源免费框架，提供 JavaScript 和 TypeScript 双重支持

## 用 canvas 画一个圆

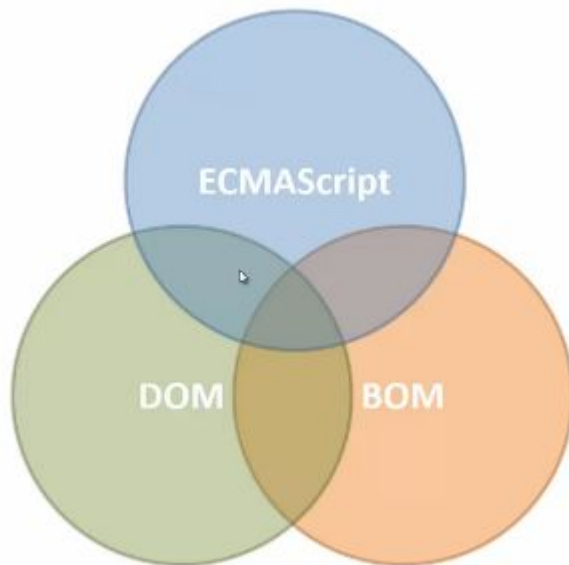
在一个 300\*300 的 canvas (id 为“myCanvas”) 上，以坐标点 (150, 150) 为圆心，100 为半径，画一个边框色为#4d4e53，填充色为#6a83ff 的圆。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>canvas 画圆</title>
</head>
<body>
  <canvas id="myCanvas" width="300" height="300"></canvas>
  <script>
    var canvas = document.getElementById('myCanvas');
    var ctx = canvas.getContext("2d");
    ctx.arc(150, 150, 100, 0, Math.PI*2,true);
    ctx.strokeStyle = "#4d4e53";
    ctx.fillStyle = "#6a83ff";
    ctx.stroke();
    ctx.fill();
  </script>
</body>
</html>
```



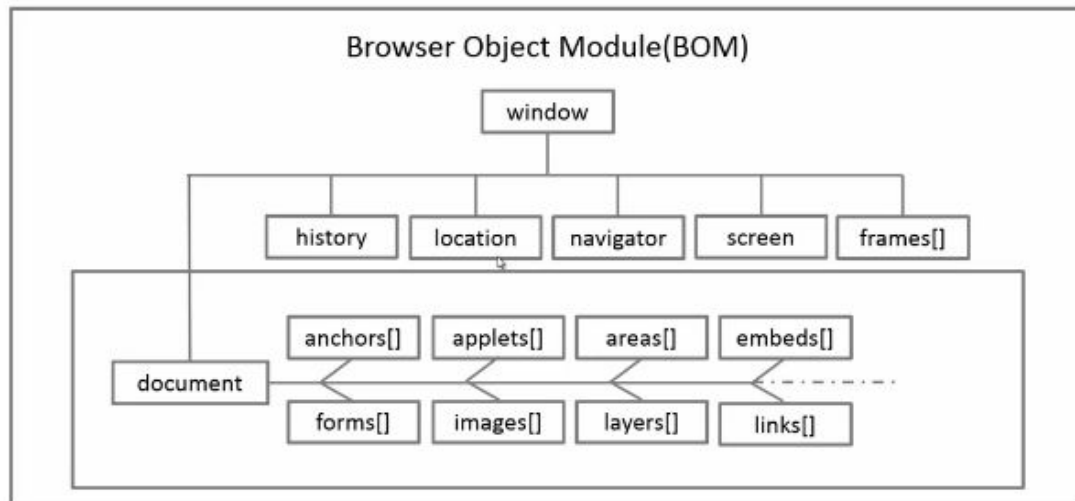
## BOM

广义的 JS 包含三部分：



BOM 为浏览器窗口对象的一组 API.

## BOM 结构图



## BOM 属性：

属性名	描述
navigator	浏览器信息
location	浏览器定位和导航
history	窗口浏览器历史
screen	屏幕信息

本笔记由西风潇潇编写，欢迎浏览博客访问更多内容：<http://www.xifengxx.com>

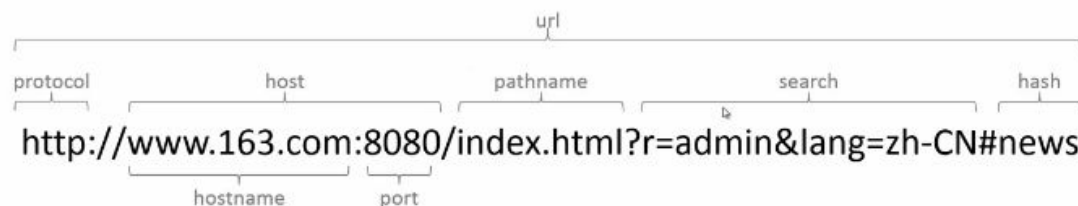
## **navigator.userAgent**

可以通过 `navigator.userAgent` 判断当前设备运行在什么浏览器上，其包含信息：

- **chrome**
  - Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.115 Safari/537.36
- **firefox**
  - Mozilla/5.0 (Windows NT 6.1; WOW64; rv:36.0) Gecko/20100101 Firefox/36.0
- **IE**
  - Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E; InfoPath.3; rv:11.0) like Gecko

## **location：浏览器定位和导航**

`location` 最重要属性为 `href`，它代表浏览器访问当前资源的完整路径，可以通过修改 `url` 属性进行浏览器跳转。



### **location 属性的方法：**

- `assign(url)` 载入新的 `url`，记录浏览记录
- `replace(url)` 载入新的 `url` 不记录浏览记录
- `reload()` 重新载入当前页

### **history 属性：**

浏览器当前窗口的浏览历史，其包含的方法有：

- `back()` 后退
- `forward()` 前进
- `go()` 正数向前，负数向后

### **Screen 属性：屏幕信息**

```
▼ screen: Screen
  availHeight: 1010
  availLeft: 0
  availTop: 0
  availWidth: 1680
  colorDepth: 24
  height: 1050
  ► orientation: ScreenOrientation
  pixelDepth: 24
  width: 1680
  ► __proto__: Screen
```

## Window 对象（BOM）下的方法：

方法名	描述
alert(),confirm(),prompt()	三种对话框
setTimeout(),setInterval()	计时器
open(),close()	开新窗口，关闭窗口

### 对话框：

- alert()
- confirm(): 返回布尔值
- prompt(): 返回用户输入内容

三种对话框运行时，都会阻塞浏览器当前线程。

### 打开或关闭窗口：open()/close()

```
var w = window.open('subwindow.html', 'subwin', 'width=300, height=300, status=yes,
resizable=yes');
// 既可关闭窗口
w.close();
```

## Window 对象事件

属性名	描述
load	文档和所有图片加载完毕时
unload	离开当前文档时
beforeunload	和unload类似，但它提供询问用户是否确定离开的机会
resize	拖动改变浏览器窗口大小时
scroll	拖动滚动浏览器时

## 表单操作

表单为页面的主要组成部分，其中包含许多的表单控件。用户通过控件与表单进行交互，提供数据并提交给服务器，服务器则做出相应的处理。

编写表单包括三个部分：

1. 构建表单
2. 服务器处理（提供接受数据接口）
3. 配置表单



### 构建表单：

以“披萨预订表单”为例来介绍

### 披萨预定表单

姓名:

电话:

邮箱:

披萨大小

☐ 小 ☐ 中 ☐ 大

披萨配料

☐ 熏肉 ☐ 奶酪 ☐ 洋葱 ☐ 蘑菇

配送时间:

```
<form>  
<p><label>姓名: <input type="text"/></label></p>  
<p><label>电话: <input type="tel"/></label></p>  
<p><label>邮箱: <input type="email"/></label></p>  
<fieldset>  
<legend> 披萨大小 </legend>  
<label> <input type="radio" name="size"/> 小 </label>  
<label> <input type="radio" name="size"/> 中 </label>  
<label> <input type="radio" name="size"/> 大 </label>  
</fieldset>  
<fieldset>  
<legend> 披萨配料 </legend>  
<label> <input type="checkbox"/> 熏肉 </label>  
<label> <input type="checkbox"/> 奶酪 </label>  
<label> <input type="checkbox"/> 洋葱 </label>  
<label> <input type="checkbox"/> 蘑菇 </label>  
</fieldset>  
<p><label>配送时间: <input type="time" value="12:45" min="11:00" max="21:00" step="900"/></label></p>  
<p><button type="button" value="提交订单"/></p>  
</form>
```

### 服务器处理

本笔记由西风潇潇编写，欢迎浏览博客访问更多内容：<http://www.xifengxx.com>

提供接口来处理用户提交数据。服务器处理的基本信息包括：

- `https://pizza.example.com/order`：服务器端提供接口地址
- `application/x-www-form-urlencoded`：服务器端接收数据使用 url 方式编码
- `custname`、`custtel`、`custemail`、`size`、`topping`、`delivery`：服务器端接收参数信息。

## 配置表单

```
<form method="post"
      action="https://pizza.example.com/order"
      enctype="application/x-www-form-urlencoded">
  <p><label>姓名: <input name="custname"></label></p>
  <p><label>电话: <input type="tel" name="custtel"></label></p>
  <p><label>邮箱: <input type="email" name="custemail"></label></p>
  <fieldset>
    <legend> 披萨大小 </legend>
    <label> <input type="radio" name="size" value="small"> 小 </label>
    <label> <input type="radio" name="size" value="medium"> 中 </label>
    <label> <input type="radio" name="size" value="large"> 大 </label>
  </fieldset>
  <fieldset>
    <legend> 披萨配料 </legend>
    <label> <input type="checkbox" name="topping" value="bacon"> 熏肉 </label>
    <label> <input type="checkbox" name="topping" value="cheese"> 奶酪 </label>
    <label> <input type="checkbox" name="topping" value="onion"> 洋葱 </label>
    <label> <input type="checkbox" name="topping" value="mushroom"> 蘑菇 </label>
  </fieldset>
  <p><label>配送时间: <input type="time" name="delivery" min="11:00" max="21:00" step="900"></label></p>
  <p><button>提交订单</button></p>
</form>
```

## 验证表单

通过在元素上增加“required”属性来强制用户填写相应信息：

```
<form method="post"
      action="https://pizza.example.com/order"
      enctype="application/x-www-form-urlencoded">
  <p><label>姓名: <input name="custname" required></label></p>
  <p><label>电话: <input type="tel" name="custtel" required></label></p>
  <p><label>邮箱: <input type="email" name="custemail"></label></p>
  <fieldset>
    <legend> 披萨大小 </legend>
    <label> <input type="radio" name="size" value="small" required> 小 </label>
    <label> <input type="radio" name="size" value="medium" required> 中 </label>
    <label> <input type="radio" name="size" value="large" required> 大 </label>
  </fieldset>
  <fieldset>
    <legend> 披萨配料 </legend>
    <label> <input type="checkbox" name="topping" value="bacon"> 熏肉 </label>
    <label> <input type="checkbox" name="topping" value="cheese"> 奶酪 </label>
    <label> <input type="checkbox" name="topping" value="onion"> 洋葱 </label>
    <label> <input type="checkbox" name="topping" value="mushroom"> 蘑菇 </label>
  </fieldset>
  <p><label>配送时间: <input type="time" name="delivery" min="11:00" max="21:00" step="900"></label></p>
  <p><button>提交订单</button></p>
</form>
```

披萨预定表单

姓名:

电话:  ! 请填写此字段。

邮箱:

披萨大小

☐ 小 ☐ 中 ☐ 大

披萨配料

☐ 熏肉 ☐ 奶酪 ☐ 洋葱 ☐ 蘑菇

配送时间:

## Form 操作内容

- 元素
- 验证
- 提交

## 元素

### form 元素

**pizzaForm.**

noValidate	true
target	abc
method	post
acceptCharset	utf-8
action	http://pizza.example.com/order
enctype/encoding	application/x-www-form-urlencoded
name	pizza
autocomplete	off

**表单提交**

```
<form novalidate
      name="pizza"
      target="abc"
      method="post"
      autocomplete="off"
      accept-charset="utf-8"
      action="http://pizza.example.com/order"
      enctype="application/x-www-form-urlencoded">
```

form 元素属性包括：

**name** 属性：获取表单节点元素

```
var pizzaForm = document.forms.pizza;
```

### autocomplete

- 有两个值 `on` 与 `off`，在设置为 `on` 时，可以自动对输入框进行补全（之前提交过的输入值），如下图：

**on**

**披萨预定表单**

姓名：

电话：

邮箱：

披萨大小：☐ 小 ☐ 中 ☐ 大

**off**

**披萨预定表单**

姓名：

电话：

邮箱：

披萨大小：☐ 小 ☐ 中 ☐ 大

- 如果在已经设置 `autocomplete="off"` 时依然出现提示框，大多数情况为浏览器设



置的自动补全（可以强制关闭）。

## elements 属性

- ① 该表单的子表单控件（除图标按钮外 `<input type="image">`）：如 `button`、`fieldset`、`input`、`keygen`、`object`、`output`、`select`、`textarea`
- ② 归属于该表单的表单控件（除图片按钮）

```
<form id="f">
  <p><label><input name="a"></label></p>
  <p><select name="b"><option>1</option></select></p>
</form>

<p><label><input name="c"></label></p>
<p><label><input name="d" form="f"></label></p>
```

elements 集合为动态节点集合

## length 属性：

等价于 `elements.length` 来用于描述表单内节点集合的个数。

## 如何选取表单空间元素：

```
<form name="test">
  <input name="a"/>
  <input name="b"/>
</form>
```

`testForm.elements[0]`

`testForm.elements['a']`

`testForm[0]`

`testForm['a']`

`form[name]` 通过名称作为索引获取表单元素时有如下特点：

- 返回 `id` 或者 `name` 为指定名称的表单空间（图片按键除外）



- 如果结果为空，则返回 `id` 为指定名称的 `img` 元素

```
<form name="test">
  
  <input name="a"/>
</form>
```

```
testForm['a']
↓
<input name="a"/>
```

```
<form name="test">
  
  <input name="a"/>
</form>
```

```
testForm['a']
↓

```

- 如果有多个同名元素，则返回这些元素的动态节点集合
- 一旦用指定名称取过该元素，则不管该元素的 `id` 或者 `name` 如何变化，只要节点还在页面上均可使用原名称获取该元素。

```
<form name="test">
  <input name="a"/>
</form>
```

1 `testForm['a'];`  
2 `testForm.elements['a'];`  
`testForm['a'].name = 'b';`

```
<form name="test">
  <input name="b"/>
</form>
```

1 `testForm['a'];`  
2 `testForm.elements['a'];`  
`testForm['a'].name = 'b';`

```
testForm['a']
↓
<input name="b"/>
```

```
testForm.elements['a']
↓
null
```

**form** 元素接口包括：

- `reset()`
- `submit()`
- `checkValidity()`

**reset()**

- 可重置元素：`input`,`keygen`,`output`,`slect`,`textarea`
- 触发表单 **reset** 事件，阻止该事件的默认行为来取消重置操作
- 元素重置时不再触发元素上的 `change` 和 `input` 事件



## label 元素

```
<label for="txtId" form="formId">
```

# label.

htmlFor	txtId
control	HTMLElement#txtId
form	HTMLFormElement#formId

### htmlFor 属性：

- 关联表单控件的激活行为（可使点击 label 与点击表单控件的行为一致）
- 可关联元素：button,input(除 hidden 外)，keygen, meter,output, progress, select,textarea



### control 属性

- 如果指定了 for 属性，则为该 for 属性对应 ID 的可关联元素
- 如果没有指定 for 属性,则为第一个子孙可关联元素。

```
<label for="txtId">文字

```

- ① 指定了for属性
- ② for属性对应ID的元素span为非可关联元素
- ③ label.control → null

### form 属性

- 关联归属表单
- 可关联元素：button/fieldset,input,keygen,label,object, output,select, textarea
- 只读属性，不可在程序中修改

可通过 label.setAttribute( 'form' ; newFormID' )来修改。

## input 元素

### type 属性

- 控件外观
- 数据类型
- 默认为 text

Keyword	State	Data type
hidden	<a href="#">Hidden</a>	An arbitrary string
text	<a href="#">Text</a>	Text with no line breaks
search	<a href="#">Search</a>	Text with no line breaks
tel	<a href="#">Telephone</a>	Text with no line breaks
url	<a href="#">URL</a>	An absolute URL
email	<a href="#">E-mail</a>	An e-mail address or list of e-mail addresses
password	<a href="#">Password</a>	Text with no line breaks (sensitive information)
date	<a href="#">Date</a>	A date (year, month, day) with no time zone
time	<a href="#">Time</a>	A time (hour, minute, seconds, fractional seconds)
number	<a href="#">Number</a>	A numerical value
range	<a href="#">Range</a>	A numerical value, with the extra semantic that th important
color	<a href="#">Color</a>	An sRGB color with 8-bit red, green, and blue comp
checkbox	<a href="#">Checkbox</a>	A set of zero or more values from a predefined lis
radio	<a href="#">Radio Button</a>	An enumerated value
file	<a href="#">File Upload</a>	Zero or more files each with a <a href="#">MIME type</a> and optio
submit	<a href="#">Submit Button</a>	An enumerated value, with the extra semantic that value selected and initiates form submission
image	<a href="#">Image Button</a>	A coordinate, relative to a particular image's siz semantic that it must be the last value selected a submission
reset	<a href="#">Reset Button</a>	n/a
button	<a href="#">Button</a>	n/a

	Control type
Hidden:	
Text:	<input type="text" value="test"/>
Search:	<input type="text" value="search"/>
Telephone:	<input type="text" value="15988898406"/>
URL:	<input type="text" value="http://test.163.com/"/>
Email:	<input type="text" value="genify@163.com"/>
Password:	<input type="password" value="....."/>
Date:	<input type="text" value="2015/03/03"/>
Time:	<input type="text" value="02:58"/>
Number:	<input type="text" value="4"/>
Range:	<input type="range" value="50"/>
Color:	<input type="color" value="#ff0000"/>
Checkbox:	<input type="checkbox"/>
Radio Button:	<input type="radio"/>
File Upload:	<input type="button" value="选择文件"/> 未选择任何文件
Submit Button:	<input type="button" value="Submit"/>
Image Button:	<input type="button" value="提交"/>
Reset Button:	<input type="button" value="Reset"/>
Button:	<input type="button" value="Button"/>

### 本地图片预览案例：



实现上面的案例，涉及到 4 个技术点：

- onchange
- accept :可接受的值有：audio/\*;video/\*;image/\*;不带“;”的 MIME type; 以“.”开始的文件后缀名
- multiple
- files

```
<input type="file" accept="image/*" multiple/>
```

```
file.addEventListener(  
    'change',function(event){  
        var files = Array.prototype.slice.call(  
            event.target.files,0  
        );  
        files.forEach(function(item){  
            file2dataurl(item,function(url){  
                var image = new Image();  
                parent.appendChild(image);  
                image.src = url;  
            });  
        });  
    });  
);
```

### select 元素



```
<select name="course">
  <option>课程选择</option>
  <optgroup label="1. DOM基础">
    <option value="1.1">1.1 文档树</option>
    <option value="1.2">1.2 节点操作</option>
    <option value="1.3">1.3 元素遍历</option>
    <option value="1.4">1.4 样式操作</option>
    <option value="1.5">1.5 属性操作</option>
    <option value="1.6">1.6 表单操作</option>
  </optgroup>
  <optgroup label="2. 事件模型">
    <option value="2.1">2.1 事件类型</option>
    <option value="2.2">2.2 事件模型</option>
    <option value="2.3">2.3 事件应用</option>
  </optgroup>
</select>
```

`select` 具有的属性和（接口）方法如下：

- name: 表单名称
- value: 第一个选中的选项的 value 值
- multiple: 多选
- options (动态节点集合)
- selectedOptions (所有已经选中的选项集合，动态节点集合)
- selectedIndex: 第一个选中选项的索引值，没有元素选中的话，返回-1.
- add(element[, before]): 在指定的位置之前添加选项用，无指定参照物则在最后添加选项。
- remove([index]): 删除某个选项

`optgroup` 所具有的属性和方法：

- disabled (分组选项不可选)
- label (分组说明)

`option` 所具有的属性和方法：

- disabled
- label (描述信息)
- value (提交表单时的数据信息)
- text (显示在页面上用户看到的文字)
- index: 当前选项的索引值
- selected
- defaultSelected: 默认情况下选项是否选中

## select 选项操作

### 创建选项

- document.createElement('option')
- new Option([text[, value[, defaultSelected[, selected]]]])





### 添加选项

- insertBefore
- select.add



### 删除选项

- removeChild
- select.remove

```
<select name="course">
  <option>课程选择</option>
  <optgroup label="1. DOM基础">
    <option value="1.1">1.1 文档树</option>
    <option value="1.2">1.2 节点操作</option>
    <option value="1.3">1.3 元素遍历</option>
  </optgroup>
  <optgroup label="2. 事件模型">
    <option value="2.1">2.1 事件类型</option>
    <option value="2.2">2.2 事件模型</option>
  </optgroup>
</select>
```

```
opt12.parentNode.removeChild(opt12);
```

```
select.remove(2);
```

级联下列选择器案例：



所需知识点：

- onchange
- remove
- add

```
<form name="course">
  <select name="chapter">
    <option>请选择章节目录</option>
  </select>
  <select name="section">
    <option>请选择节目目录</option>
  </select>
</form>
```

```
var chapters = [
  {text:'1. DOM基础',value:'1'},
  {text:'2. 事件模型',value:'2'}
];
var sections = {
  1:[
    {text:'1.1 文档树',value:'1.1'},
    {text:'1.2 节点操作',value:'1.2'},
    {text:'1.3 元素遍历',value:'1.3'},
    {text:'1.4 样式操作',value:'1.4'},
    {text:'1.5 属性操作',value:'1.5'},
    {text:'1.6 表单操作',value:'1.6'}
  ],
  2:[
    {text:'2.1 事件类型',value:'2.1'},
    {text:'2.2 事件模型',value:'2.2'},
    {text:'2.3 事件应用',value:'2.3'}
  ]
};
```

```
function fillSelect(select,list){
  for(var i=select.length-1;i>0;i--){
    select.remove(i);
  }
  list.forEach(function(data){
    var option = new Option(data.text,data.value);
    select.add(option);
  });
}
fillSelect(chapterSelect,chapters);
chapterSelect.addEventListener(
  'change',function(event){
    var value = event.target.value,
        list = sections[value]||[];
    fillSelect(sectionSelect,list);
  }
);
```

## textarea 元素

textarea 具有的属性和方法如下：

- name
- value （用户输入信息）
- select() （全选当前输入的内容）
- selectionStart （选中的内容的起始位置，无选中时返回当前光标所在位置）
- selectionEnd （选中内容结束位置，无选中时返回光标位置）
- selectionDirection （返回选择的方向 forward backward）
- setSelectionRange(start, end[, direction]) （使用程序选中内容）
- setRangeText(replacement[, start, end, [mode]]) （设置内容范围）

## selection :

表示选择区域，对于 input 元素同样有效。





## ● meter

主流的浏览器对表单元素的支持情况:

<https://html5test.com/compare/browser/ie-8/ie-9/ie-10.html> (链接已失效)

### 如何处理由于浏览器设置导致的 autocomplete="off" 失效的问题

如何处理由于浏览器设置导致的 autocomplete="off" 失效的问题，给出兼容各浏览器的解决方案

1. 移除 input 的 id 和 name; 浏览器的自动补全功能依赖于 id 或者 name 来识别各种框框，找不到 id 和 name 它就不认识这个框了，于是放弃

2. 设置 autocomplete="随机的非法值"; 将该属性赋一个除 on 或 off 之外的一个随便的值, 这样浏览器会放弃对该属性的执行。如:

```
<input name="a" autocomplete="nope">
```

3. 在密码输入框前加一个隐藏的 input [type="password"] 元素。(让这个隐藏的框在真正的框前挺身而出，替它挡住来自浏览器的一枪; 浏览器一般也开一枪，干掉这个替死鬼之后就撤了) 如:

```
<input type="password" style="display:none">
  <input type="password" name="b"><button>Sign in</button></form>
```

4. 先将 type="password" 的表单域设置 type="text", 聚焦时再改为 type="password"

```
<input type="password" onfocus="this.setAttribute('type','text')"
onblur="this.setAttribute('type','password')">
```

### 封装一个输入框通用的光标操作接口

封装一个输入框通用的光标操作接口，使得在传入输入框和光标的起始、结束位置后，可以将输入框中从起始位置到结束位置的内容选中，如

```
function selection(input, start, end) {
  // input 为输入框，如 input、textarea
  // start 为光标起始位置，如 0
  // end 为光标结束为止，如 10
  // TODO
```

代码如下:

```
function selection(input, start, end) {
  // input 为输入框，如 input、textarea
  // start 为光标起始位置，如 0
  // end 为光标结束为止，如 10
  if(input.setSelectionRange) { //非 IE
    input.setSelectionRange(start, end); //从第 start 个字到第 end 个字
    input.focus(); //光标聚焦到 input 上
  }
  else if(input.createTextRange) { //IE
    var range=input.createTextRange(); //在 input 下创建一个文本选择
    range.collapse(true); //将光标定位到起始点
    range.moveStart('character', start); //光标移动到第 start 个字前
    range.moveEnd('character', end-start); //此方法是从 start 位置选择几个字，
    因此需要用结束位置减去开始位置
```

```
range.select();//IE 下光标聚焦到 input 上
}
```

## 表单验证

### 验证元素

可验证元素：button, input, select, textarea

以下情况不做验证：

- input 元素，type 类型为 hidden, reset, button 时
- button 元素，type 类型为 reset, button 时
- input /textarea, 当属性为 readonly 时
- 所有可验证元素作为 datalist 的子孙节点时
- 当元素被禁用时 disabled 的状态

验证涉及到的属性和接口：

- willValidate （表明此元素在表单提交时是否会被验证）
- checkValidity() （用于验证元素，返回 true 当验证通过，否则触发 invalid 事件）
- validity （存储验证结果，不同的异常状态都会在这个属性里标明出来）
- validationMessage （显示验证异常信息）
- setCustomValidity(message) （自定义验证错误信息）

**validity**：存储验证过程中可能涉及到的错误信息，包含以下内容：

名称	描述
valueMissing	设置了required没有value
typeMismatch	value与type不符，如email, url
patternMismatch	value与pattern不匹配
tooLong	value长度超过maxlength指定的长度
tooShort	value长度小于minlength指定的长度
rangeUnderflow	value值小于min指定的值
rangeOverflow	value值大于max指定的值
stepMismatch	value值不符合step指定的值
badInput	输入不完整
customError	使用setCustomValidity设置了自定义错误
valid	符合验证条件

自定义异常：

涉及的点包括

- oninvalid
- setCustomValidity

```
<form action="./api" method="post">
  <p><label>姓名: <input name="username" required/></label></p>
  <p><button>submit</button></p>
</form>
```



```
input.addEventListener(
  'invalid',function(event){
    var target = event.target;
    if (target.validity.valueMissing){
      target.setCustomValidity('请输入姓名! ');
    }
  }
);
```

禁止验证:



```
<form action="./api" method="post" novalidate>
  <label>电话: <input type="number" name="tel"/></label>
  <button>提交</button>
</form>
```

如上代码, `input` 类型为 `number`, 当输入时, 会弹出数字键盘来做验证。这时我们需要其不对输入的内容进行验证, 可以通过在表单里设置 `novalidate` 属性来禁止其验证。

## 表单提交

### 隐式提交

- 如: 聚焦在输入框时按回车提交表单
- 满足以下任一条件
  - 表单有非禁用的提交按键

```
<form action="/api">
  <input name="a"/>
  <input name="b"/>
  <input name="c"/>
  <button>submit</button>
</form>
```

，此时 a/b/c 聚焦输入时按下回车键，即可提交表单。

- 没有提交按键时，不超过一个类型为 text search url email password date time number 的 input 元素

```
<form action="/api">
  <input name="a"/>
  <input name="b"/>
  <input name="c"/>
  <button>submit</button>
</form>
```

，此时在输入框里输入后，按回车键，也可以隐式

的提交表单。

## 提交过程

- 根据表单 enctype 指定的值构建要提交的数据结构
- 使用 method 指定的方式发送数据到 action 指定的目标。

## 构建提交数据

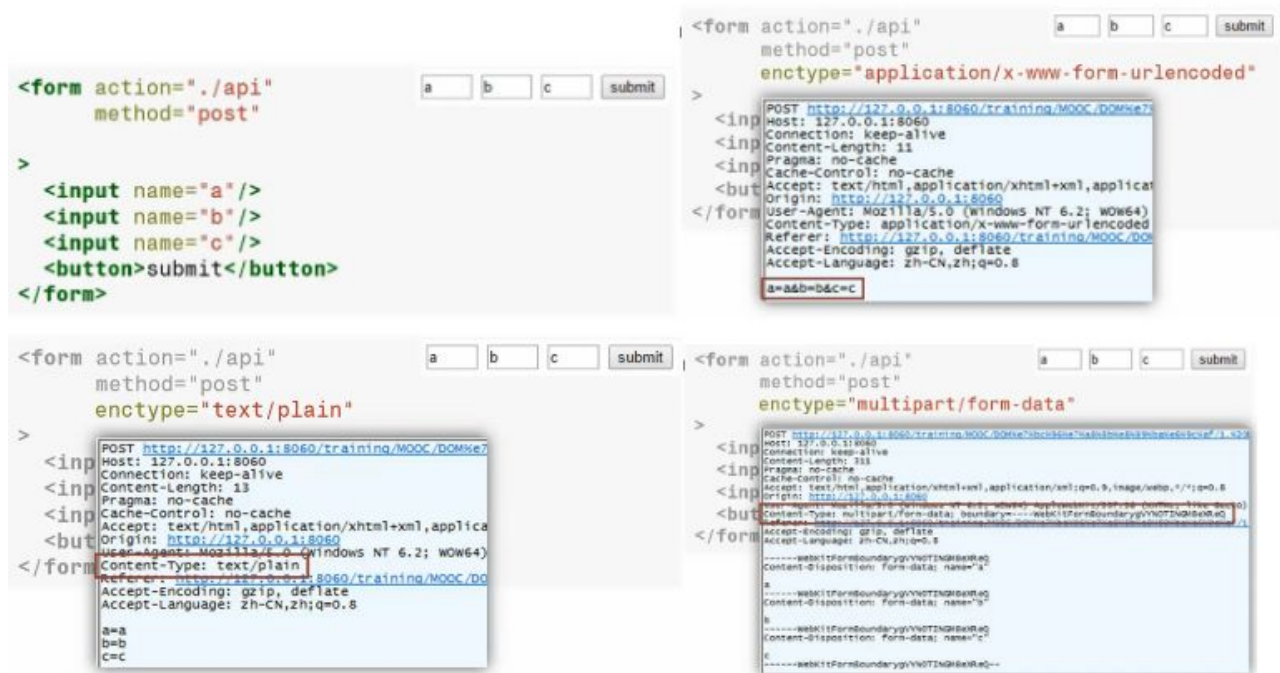
- 从可提交元素中提取数据组成指定的数据结构过程
- 可提交元素有 button input keygen object select textarea

## 编码方式 ( enctype )

enctype 所支持的形式：

- application/x-www-form-urlencoded （默认，数据格式为 & 分隔的键值对）
- multipart/form-data （RFC 2388 字节流形式，例如文件上传所使用的数据编码形式）
- text/plain （回车换行符分隔的键值对）

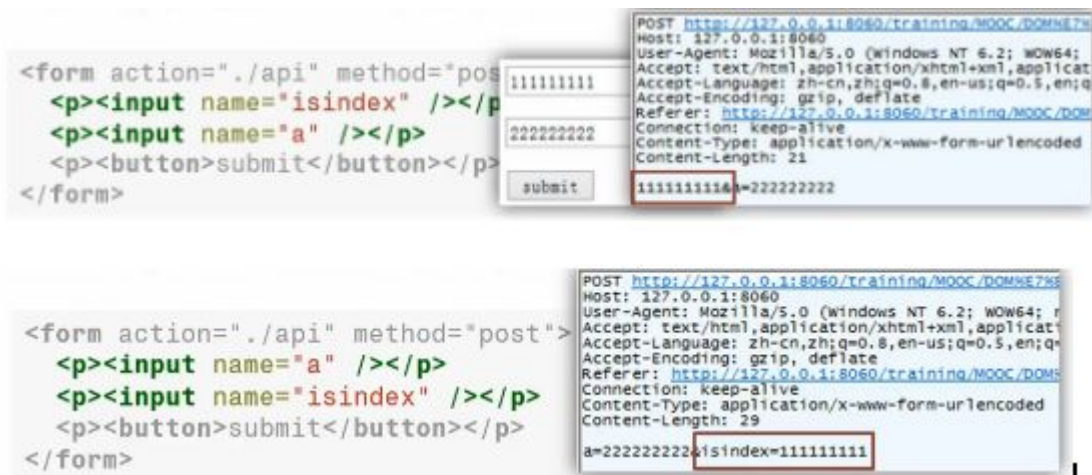




## 特殊案例

1. 当一个表单元素 `name="isindex"` 并且 `type="text"` 而且满足如下要求时：

- 编码格式为 `application/x-www-form-urlencoded`
- 作为表单的第一个元素
- 则提交时只发送 `value` 值，不包含 `name`。



2. 当 `name="_charset_"` 并且类型为 `type="hidden"` 时，而且满足如下要求时：

- 没有设置 `value` 值
- 提交时 `value` 自动使用当前提交的字符集填充。



表单提交接口：

### submit()

- 提交表单：form.submit();

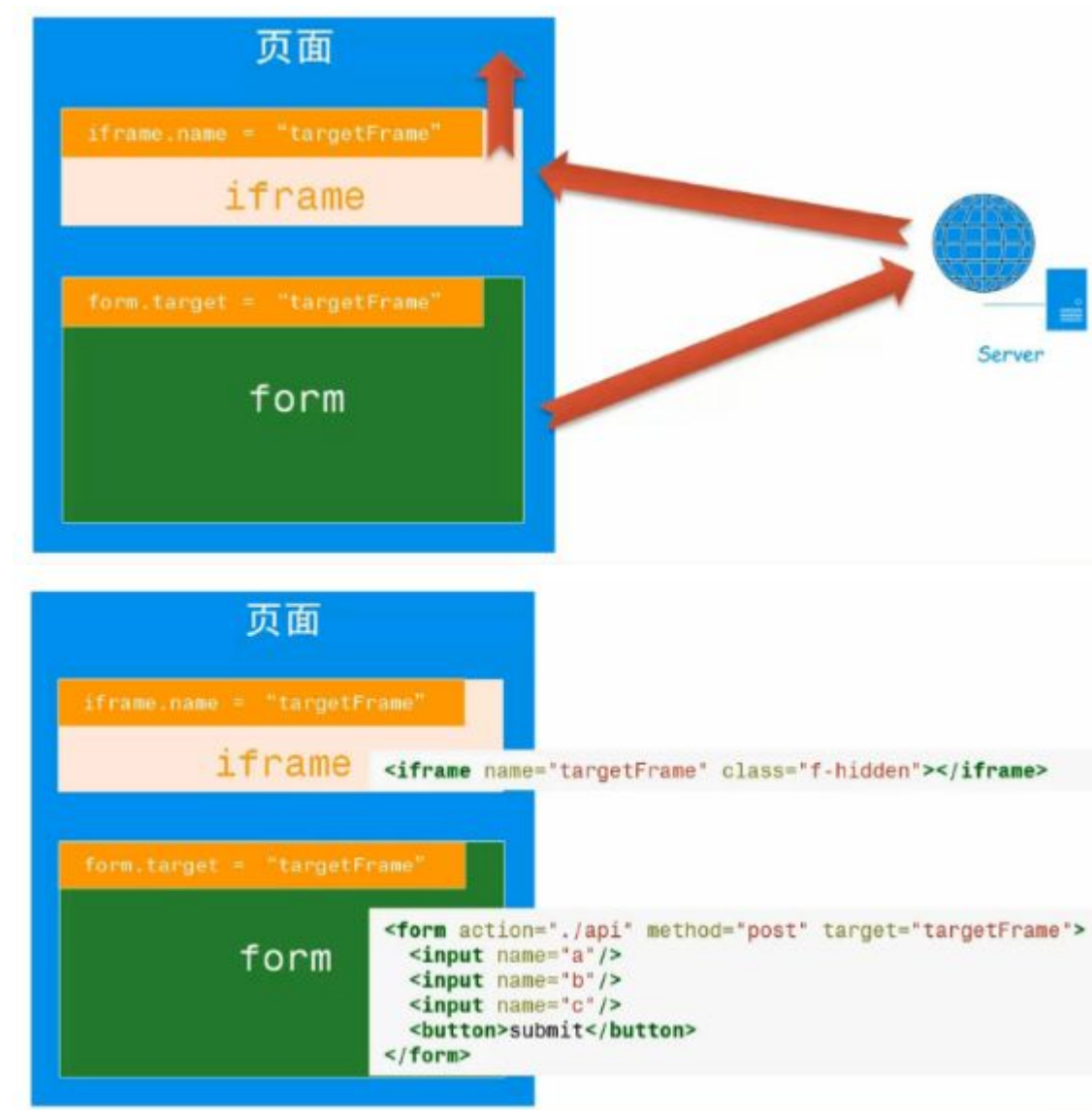
### onsubmit()

- 表单提交事件
- 提交之前的数据验证
- 阻止事件的默认行为来取消表单提交

```
form.addEventListener(  
    'submit',function(event){  
        var notValid = false;  
        var elements = event.target.elements;  
  
        // TODO 处理自定义的验证规则  
  
        if (notValid){  
            event.preventDefault();  
        }  
    }  
);
```

无刷新表单提交范例：

常用的方式是通过 AJAX 进行实现，这里使用 iframe 来做中介代理实现。



利用 `iframe` 实现表单的无刷新提交，兼容到主流浏览器（IE6+，Firefox 最新，Chrome 最新）。

代码如下：

```
<!Doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>利用 iframe 实现表单的无刷新提交</title>
  </head>
  <body>
    <form target="formtarget" method="post">
      <input type="text" name="username">
      <input type="password" name="password">
      <input type="submit">
    </form>
    <iframe name="formtarget" style="display:none"></iframe>
```



```
<div id='div1'></div>

<script type="text/javascript">
    var enventUtil={
        addHandler:function(elem, type, handler) {
            if(elem.addEventListener) {
                elem.addEventListener(type, handler, false);
            }else if(elem.attachEvent) {
                elem.attachEvent('on'+type, handler);
            }else {
                elem['on'+type]=handler;
            }
        }
    }

    var oFrame=document.getElementsByTagName('iframe');
    var oDiv=document.getElementById('div1');
    enventUtil.addHandler(oFrame, 'load', function() {
        var frameDocument=oFrame.contentDocument || oFrame.contentwind
w.document;
        if(frameDocument) {
            oDiv.innerHTML=frameDocument.body.innerHTML;
        }
    })
</script>
</body>
</html>
```

## 表单应用

效果图：

手机号码登录

请输入正确的手机号码

手机号：13888888888aaa  
11位数字手机号码

密码：  
至少6位，同时包含数字和字母

登录

手机号码登录

密码必须包含数字和字母

手机号：13888888888  
11位数字手机号码

密码：.....  
至少6位，同时包含数字和字母

登录



登录接口：

- 请求地址：/api/log
- 请求参数：
  - telephone 手机号码
  - password 密码，MD5 加密
- 返回结果：
  - code 请求状态，200 表示成功
  - result 请求结果数据

```
<form action="/api/login" class="m-form" name="loginForm" target="result" autocomplete="off">
  <legend>手机号码登录</legend>
  <fieldset>
    <div class="msg" id="message"></div>
    <div>
      <label for="telephone">手机号: </label>
      <input id="telephone" name="telephone" class="u-txt"
        type="tel" maxlength="11" required
        pattern="^0?(13[0-9]|15[012356789]|18[0236789]|14[57])[0-9]{8}$"><br/>
      <span class="tip">11位数字手机号码</span>
    </div>
    <div>
      <label for="password">密 码: </label>
      <input id="password" name="password" type="password" class="u-txt"><br/>
      <span class="tip">至少6位，同时包含数字和字母</span>
    </div>
    <div><button name="loginBtn">登 录</button></div>
  </fieldset>
</form>
```

详细代码参见 demo.

## 列表操作

列表的常用形式有图片形式与信息形式两种：



列表常见的操作有：

- 显示列表
- 选择列表项
- 新增列表项
- 删除列表项
- 更新列表项

## 范例效果：

	♥	音乐标题	歌手	专辑	时长
01	♥	小鸡哔哔	群星	热门华语205	02:40
02	♥	Häschenparty	Schnuffel	Ich hab' Dich lieb	02:44
03	♥	小苹果	筷子兄弟	老男孩之猛龙过江 电影原声	03:31
04	♥	小三	冷漠	这条街	04:47
05	♥	我爱你胜过你爱我	冷漠	我爱你胜过你爱我	04:43
06	♥	看透爱情看透你	冷漠	这条街	04:38
07	♥	这条街	冷漠	这条街	03:53
08	♥	伤心城市	冷漠		04:27
09	♥	多情的人不该相遇	冷漠	次+精选	04:36
10	♥	元芳你怎么看	冷漠	元芳你怎么看[单曲]	03:59
11	♥	小三续集	冷漠	我爱你胜过你爱我	04:09
12	♥	寂寞是你给的苦	冷漠	寂寞是你给的苦	04:03

数据定义：



显示列表:

```
<div class="m-plylist m-plylist-sort s-bfc5" id="parent">
  <div class="head f-cb">
    <div class="fix">
      <div class="th col f-pr"></div>
      <div class="th col o-love">
        <span class="ico u-icn4 u-icn4-love"></span>
      </div>
    </div>
    <div class="flow f-cb">
      <div class="th col">音乐标题</div>
      <div class="th col">歌手</div>
      <div class="th col">专辑</div>
      <div class="th col">时长</div>
    </div>
  </div>
</div>

<ul>
  {list tracks as track}
  <li class="ite j-item">
    <span class="ico u-icn4 u-icn4-love"></span>
    <div class="flow f-cb">
      <div class="td col title">
        <a href="/track/${track.id}/" class="tit s-bfc8">${track.name}</a>
      </div>
      <div class="td col ellipsis">
        <a href="/artist/${track.artist.id}/" class="s-bfc8">${track.artist.name}</a>
      </div>
      <div class="td col ellipsis">
        <a href="/album/${track.album.id}/" class="s-bfc4">${track.album.name}</a>
      </div>
      <div class="td col">${track.duration|dur2str}</div>
    </div>
  </li>
</ul>
</div>
```

绘制列表:

本笔记由西风潇潇编写，欢迎浏览博客访问更多内容：<http://www.xifengxx.com>

```
function render(parent,list){
    var ext = {
        dur2str:function(duration){
            duration = duration/1000;
            var m = Math.floor(duration/60),
                s = Math.floor(duration%60);
            return (m<10?'0':'')+m+':'+(s<10?'0':'')+s;
        }
    };
    var html = Trimpath.merge(
        tplContent,{tracks:list},ext
    );
    parent.insertAdjacentHTML('beforeEnd',html);
}
```

通过 Ajax 获取列表：

```
var xhr = new XMLHttpRequest();
xhr.open('GET','/api/track',true);
xhr.onload = function(){
    render(
        document.getElementById('parent'),
        JSON.parse(xhr.responseText)
    );
};
xhr.send(null);
```

选择列表项：

单选：



```
parent.addEventListener(
    'mousedown',function(event){
        var target = getTarget(event);
        if (!!target&&!isSelected(target)&&
            !event.ctrlKey&&!event.shiftKey){
            clearSelection();
            appendToSelection(target);
        }
    }
);
```

多选：



## 右键菜单

```
function showContextMenu(selection,left,top){
    // build menu items
    var actions = [
        {text:'删除歌曲',value:'delete'}
    ];
    if (selection.length<=1){
        actions.push(
            {text:'插入歌曲',value:'insert'},
            {text:'编辑歌曲',value:'update'}
        );
    }
    // show menu
    var menu = getMenu(
        TrimPath.merge(
            tplMenu,{actions:actions}
        )
    );
    menu.style.top = top+20+'px';
    menu.style.left = left+10+'px';
    document.body.appendChild(menu);
}
```

## 增加列表项



```
function insertTrack(){
    showTrackAddForm(function(track){
        selection[0].insertAdjacentElement(
            'beforeBegin', getTrackItem(track)
        );
    });
}
```

```
var getTrackItem = function(track){
    var div = document.createElement('div');
    render(div,[track]);
    return div.getElementsByTagName('li')[0];
};
```

## 删除列表项

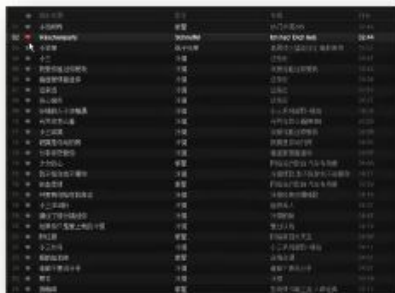


```
function removeTrack(){
    selection.forEach(function(node){
        node.parentNode.removeChild(node);
    });
    selection = [];
```

## 更新列表项

```
function updateTrack(){
    var node = selection[0];
    showTrackUpdateForm(
        node,function(track){
            var list = node.getElementsByTagName('a');
            list[0].textContent = track.name;
            list[1].textContent = track.album.name;
            list[2].textContent = track.artist.name;
        });
}
```

## 更新状态



```
parent.addEventListener(
    'mousedown',function(event){
        // love all track
        var target = getTarget(event,'j-lvbtn');
        if (!!target){
            var loved = toggleLove(target);
            loved ? loveAll() : unloveAll();
            return;
        }
        // love one track
        var target = getTarget(event,'j-love');
        if (!!target){
            toggleLove(target);
            syncLoveAllState();
        }
    });
```

## 编程方式

### 面向视图编程方式:


即针对视图的直接编程 ( 对 DOM 树进行直接的操作 )

```
// verify first name
pro.onFirstNameChange = function(){
    var firstName = getFromView('firstName');
    if (this.checkFirstName(firstName)){
        this.updateViewPass('firstName');
    }else{
        this.updateViewFailed('firstName');
    }
};


// verify last name
pro.onLastNameChange = function(){
    var lastName = getFromView('lastName');
    if (this.checkLastName(lastName)){
        this.updateViewPass('lastName');
    }else{
        this.updateViewFailed('lastName');
    }
};
```

### 面向数据编程方式






```
// view model
this.data = {
  firstName : 'a',
  lastName : 'bcdef',
  status:{
    firstName : STATUS_NOT_CHECK,
    lastName : STATUS_NOT_CHECK
  }
};
```



```
// view model
// verify first name
this.watch('firstName',function(event){
  if (self.checkFirstName(this.firstName)){
    this.status.firstName = STATUS_PASS;
  }else{
    this.status.firstName = STATUS_FAILED;
  }
});
```



```
// verify first name
// verify last name
this.watch('lastName',function(event){
  if (self.checkLastName(this.lastName)){
    this.status.lastName = STATUS_PASS;
  }else{
    this.status.lastName = STATUS_FAILED;
  }
});
```

这种操作方式，视图则被抽象为若干的数据以及状态（后续所有的操作都会更加数据模型而操作），从而做到视图模型层完全自动化的测试。