# 整体架构-jQuery3.2.1

# 创建jQuery对象----(L94)

# jQuery原型添加方法与属性

# 核心函数

# 选择器sizzle引擎

# 回调对象

# Deferred对象(异步队列)

# 数据缓存

# 队列操作

# 事件系统

# DOM操作

# 样式操作

# class操作

# 位置尺寸操作

# 属性操作

## 概述

### attribute和property分析

* + - 1. 规范中：

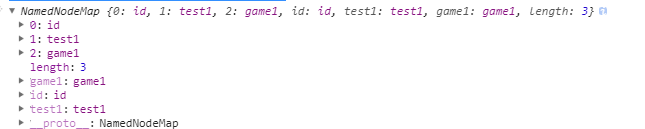
读取和设置attribute的方法是getAttribute/setAttribute/removeAttribute

访问元素的property，则需要用.(点)的方法，比如，box.somekey

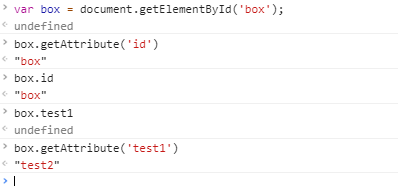
* + - 1. attribute
         1. 每个DOM元素都有一个对应的attributes属性来存放所有的attribute节点，对于如下html：

<div id="box" test1="test2" game1="gamesomeabc">box content</div>

* + - * 1. 调用document.getElementById('box').attributes;获得的结果是：



* + - * 1. Attributes数组是动态变化的
      1. property
         1. 如果把DOM元素看成是一个普通的Object对象，那么property就是一个以名值对(name=”value”)的形式存放在Object中的属性
      2. 共用property与attribute
         1. 规范中，浏览器会默认的将一些attribute与property共用，比如id,class,style等，我们可以通过getAttribute()或者点的方式进行获取
         2. 如果setAttribute的属性是浏览器中不共用的，我们就只能通过getAttribute进行获取。
         3. 针对上述html代码，因id属性是共用的，可以通过两种方式获取，但test1是自定义的html属性，只能通过getAttribute获取到值



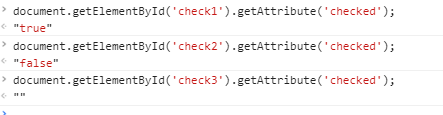
* + - 1. 注意有些共用属性名：
         1. 对某个attribute，创建了对应的property，但却用了不同的名称，比如class，对应的property为className

document.getElementsByTagName('div')[0].className; // "dddd"  
document.getElementsByTagName('div')[0]['class']; // undefined

### Checkbox的checked属性

* + - 1. **checkbox的checked属性，它的值为Boolean类型**
      2. 只要特性checked在标签里出现了，不管值是什么，复选框就会被选中。此时属性checked为true，否则为false；
      3. 后续修改特性checked的值，不会导致checkbox的选中状态改变；
      4. 后续修改属性checked的值，会导致checkbox的选中状态改变；
      5. 全部为选中状态，因此checked值并不影响是否选中的状态

<input id="check1" type="checkbox" checked="true">  
<input id="check2" type="checkbox" checked="false">  
<input id="check3" type="checkbox" checked>



* + - 1. 选中状态的表示
         1. document.getElementById('check3').checked--------------------true为选中，false为未选中
         2. 因此，通过document.getElementById('check3').checked = true，可以将未选中状态改为选中

## jQuery.attr()

### attrHandle-------------(L7593)

// jQuery.expr.match.bool.source = "^(?:checked|selected|async|autofocus|autoplay|controls|defer|disabled|hidden|ismap|loop|multiple|open|readonly|required|scoped)$"  
// 即匹配boolean属性，\w:由数字、26个英文字母或者下划线组成的字符串  
// jQuery.expr.match.bool.source.match( /\w+/g )会将字符串转换为数组形式  
jQuery.each( jQuery.expr.match.bool.source.match( /\w+/g ), function( i, name ) {  
 var getter = attrHandle[ name ] || jQuery.find.attr;  
  
 attrHandle[ name ] = function( elem, name, isXML ) {  
 var ret, handle,  
 lowercaseName = name.toLowerCase();  
  
 if ( !isXML ) {  
  
 // 通过临时删除这个函数来避免一个无限循环。  
 // 因为开始gettter默认为jQuery.find.attr =Sizzle.attr  
 // 而Sizzle.attr计算内部val时，又会调用这个函数获得val值  
 // 通过Sizzle.attr与当前attrHandle函数，对于checked等boolean属性  
 // 返回的则是 ret = lowercaseName，则是checked  
 handle = attrHandle[ lowercaseName ];  
 attrHandle[ lowercaseName ] = ret;  
 ret = getter( elem, name, isXML ) != null ?  
 lowercaseName :  
 null;  
 attrHandle[ lowercaseName ] = handle;  
 }  
 return ret;  
 };  
} );

### boolHook-------------------------(L7579)

// boolean属性的hooks  
boolHook = {  
 set: function( elem, value, name ) {  
 // 如value为false，则删除属性,否则设置属性  
 if ( value === false ) {  
 //attr("checked",false)调用将会把checked的属性变成"undefined"  
 jQuery.removeAttr( elem, name );  
 } else {  
 elem.setAttribute( name, name );  
 }  
 return name;  
 }  
};

### attrHooks.type.set-----------------------(L7547)

attrHooks: {  
 type: {  
 set: function( elem, value ) {  
 // 针对IE <=11 only，value为radio，并且元素是input  
 if ( !support.radioValue && value === "radio" &&  
 nodeName( elem, "input" ) ) {  
 var val = elem.value;// 复制原值，设置属性，然后将原值设置回去  
 elem.setAttribute( "type", value );  
 if ( val ) {  
 elem.value = val;  
 }  
 return value;  
 }  
 }  
 }  
},

### 源码解析----------------(L7501)

attr: function( elem, name, value ) {  
 var ret, hooks,  
 nType = elem.nodeType;  
  
 // 不要在text, comment and attribute 节点上get/set attributes  
 if ( nType === 3 || nType === 8 || nType === 2 ) {  
 return;  
 }  
  
 // 如getAttribute方法不支持，则使用prop方法获取  
 if ( typeof elem.getAttribute === "undefined" ) {  
 return jQuery.prop( elem, name, value );  
 }  
  
 // Attribute hooks 使用name的全小写key  
 // 1、一些特殊的特性，如href、width等=>attrHooks  
 // 2、一些值为Boolean的属性，如checked等=>boolHook  
 if ( nType !== 1 || !jQuery.isXMLDoc( elem ) ) {  
 hooks = jQuery.attrHooks[ name.toLowerCase() ] ||  
 // 对于类似checked属性，利用boolHook  
 // boolHook只有hooks.set( elem, value, name ),对于attr('checked',false),则是删除属性，其他的为设置属性  
 ( jQuery.expr.match.bool.test( name ) ? boolHook : undefined );  
 }  
  
 if ( value !== undefined ) {  
 // value === null 则删除name  
 if ( value === null ) {  
 jQuery.removeAttr( elem, name );  
 return;  
 }  
 // 如hooks有set方法，则调用set，设置属性值  
 if ( hooks && "set" in hooks &&  
 ( ret = hooks.set( elem, value, name ) ) !== undefined ) {  
 return ret;  
 }  
 // 否则，直接调用setAttribute方法设置属性值  
 elem.setAttribute( name, value + "" );  
 return value;  
 }  
 // 如val为undefined，且有hooks的get方法，直接获取，并且返回值ret避免==null  
 if ( hooks && "get" in hooks && ( ret = hooks.get( elem, name ) ) !== null ) {  
 return ret;  
 }  
 // 为了保证getAttribute兼容性，调用sizzle里面的attr方法获取属性值  
 ret = jQuery.find.attr( elem, name );  
  
 // 对于不存在的属性，调用此方法，返回null，转换为undefined  
 return ret == null ? undefined : ret;  
},

## jQuery.removeAttr()-----------(L7563)

/\*\*  
 \* 删除属性，本质还是调用removeAttribute  
 \* @param elem 要删除的元素  
 \* @param value value要删除的属性名，可以是'class test'  
 \* 因此，可以这样一起删除多个属性jQuery.removeAttr(document.getElementById('p1'),'class test')  
 \*/  
removeAttr: function( elem, value ) {  
 var name,  
 i = 0,  
  
 // 属性名能包含非html空白字符  
 // https://html.spec.whatwg.org/multipage/syntax.html#attributes-2  
 attrNames = value && value.match( rnothtmlwhite );  
  
 if ( attrNames && elem.nodeType === 1 ) {  
 while ( ( name = attrNames[ i++ ] ) ) {  
 elem.removeAttribute( name );  
 }  
 }  
}

## $().attr(),$().removeAttr()------------(L7488)

jQuery.fn.extend( {  
 /\*\*  
 \* 获取匹配集合第一个元素的属性值；设置一个或多个属性给所有匹配集合  
 \* attr使用access包装  
 \* 它拥有了access提供的四种传值方式：name，(name,value)，({key:value,key:value})，(name,function(index,attr){})  
 \* @param name  
 \* @param value  
 \* @return {\*}  
 \*/  
 attr: function( name, value ) {  
 return access( this, jQuery.attr, name, value, arguments.length > 1 );  
 },  
  
 removeAttr: function( name ) {  
 return this.each( function() {  
 jQuery.removeAttr( this, name );  
 } );  
 }  
} );

## jQuery.prop()

### propHooks----------------(L7665)

propHooks: {  
 tabIndex: {  
 get: function( elem ) {  
  
 // Support: IE <=9 - 11 only  
 // elem.tabIndex当没有明确设置时，并不能总是返回正确值  
 // https://web.archive.org/web/20141116233347/http://fluidproject.org/blog/2008/01/09/getting-setting-and-removing-tabindex-values-with-javascript/  
 // 使用适当的attribute检索(#12072)  
 // tabindex 属性规定元素的 tab 键控制次序  
 // 以下元素支持 tabindex 属性：<a>, <area>, <button>, <input>, <object>, <select> 以及 <textarea>。  
 var tabindex = jQuery.find.attr( elem, "tabindex" );  
  
 if ( tabindex ) {  
 return parseInt( tabindex, 10 );  
 }  
 // &&优先级高于||，因此是elem具有href的a或area，或input等元素，返回0，否则返回-1  
 if (  
 rfocusable.test( elem.nodeName ) ||  
 rclickable.test( elem.nodeName ) &&  
 elem.href  
 ) {  
 return 0;  
 }  
  
 return -1;  
 }  
 }  
},

### propFix----------------------(L7693)

propFix: {  
 "for": "htmlFor",  
 // 概述中也介绍  
 // document.getElementsByTagName('div')[0].className; // "dddd"  
 // document.getElementsByTagName('div')[0]['class']; // undefined  
 // 因此需要对特殊的name进行转换才能获得准确值  
 "class": "className"  
}

### 源码解析--------------------------(L7633)

// 整体逻辑与attr类似，只是获取值的与设置值的方式是 elem[ name ] = value 与 elem[ name ]  
prop: function( elem, name, value ) {  
 var ret, hooks,  
 nType = elem.nodeType;  
  
 // // 不要在text, comment and attribute 节点上get/set attributes  
 if ( nType === 3 || nType === 8 || nType === 2 ) {  
 return;  
 }  
  
 if ( nType !== 1 || !jQuery.isXMLDoc( elem ) ) {  
  
 // 修复名字，添加钩子  
 name = jQuery.propFix[ name ] || name;  
 hooks = jQuery.propHooks[ name ];  
 }  
  
 if ( value !== undefined ) {  
 if ( hooks && "set" in hooks &&  
 ( ret = hooks.set( elem, value, name ) ) !== undefined ) {  
 return ret;  
 }  
 // prop是利用elem[name] =value 方式赋值  
 return ( elem[ name ] = value );  
 }  
  
 if ( hooks && "get" in hooks && ( ret = hooks.get( elem, name ) ) !== null ) {  
 return ret;  
 }  
  
 return elem[ name ];  
},

## $().prop(),$().removeProp()---------(L7620)

jQuery.fn.extend( {  
 prop: function( name, value ) {  
 return access( this, jQuery.prop, name, value, arguments.length > 1 );  
 },  
 // 删除prop，则是循环调用delete 删除，如name需要fix，则用fix后的名字  
 removeProp: function( name ) {  
 return this.each( function() {  
 delete this[ jQuery.propFix[ name ] || name ];  
 } );  
 }  
} );

## jQuery.propHooks.selected---------(L7699)

// Support: IE <=11 only  
// 获取元素option的selected属性,修复在IE默认不选中的BUG  
// eslint 规则 "no-unused-expressions" 在这段代码中设为disabled，因为这段代码可能被认为错误  
// no-unused-expressions：未使用过的表达式在程序中不起任何作用，通常是个逻辑错误  
// 例如，n + 1; 不是语法错误，但它可能是程序员的书写错误，原本是想写赋值语句 n += 1;。  
// 该问题的解决方案是在访问`selected`属性时，先访问其父级`<select>`元素的`selectedIndex`属性(selectedIndex 属性可设置或返回下拉列表中被选选项的索引号。)  
// 强迫浏览器计算`<option>`的`selected`属性，以得到正确的值。  
// 需要注意的是`<option>`元素的父元素不一定是`<select>`，也有可能是`<optgroup>`。  
if ( !support.optSelected ) {  
 jQuery.propHooks.selected = {  
 get: function( elem ) {  
  
 /\* eslint no-unused-expressions: "off" \*/  
  
 var parent = elem.parentNode;  
 if ( parent && parent.parentNode ) {  
 parent.parentNode.selectedIndex;  
 }  
 return null;  
 },  
 set: function( elem ) {  
  
 /\* eslint no-unused-expressions: "off" \*/  
  
 var parent = elem.parentNode;  
 if ( parent ) {  
 parent.selectedIndex;  
 // 确保也适用于optgroups元素,<option>父级可能是`<optgroup>`  
 if ( parent.parentNode ) {  
 parent.parentNode.selectedIndex;  
 }  
 }  
 }  
 };  
}

## jQuery.propFix.tableIndex等-------(L7735)

// 其实就是创建jQuery.propFix.tabIndex =tabIndex,jQuery.propFix.readOnly=readOnly等  
jQuery.each( [  
 "tabIndex",  
 "readOnly",  
 "maxLength",  
 "cellSpacing",  
 "cellPadding",  
 "rowSpan",  
 "colSpan",  
 "useMap",  
 "frameBorder",  
 "contentEditable"  
], function() {  
 jQuery.propFix[ this.toLowerCase() ] = this;  
} );

## $().val()

### 注意：

* + - 1. **该函数常用于设置或获取表单元素的value属性值。例如：<input>、<textarea>、<select>、<option>、<button>等**
      2. 只有在为val属性时，才会返回text文本

### ValHooks.select,valHooks.option-------------------(L7998)

jQuery.extend( {  
 valHooks: {  
 option: {  
 get: function( elem ) {  
 // 对于option，有value值应该返回value值  
 // <option value="1">One</option>如，返回应该为1,  
 var val = jQuery.find.attr( elem, "value" );  
 return val != null ?  
 val :  
  
 // Support: IE <=10 - 11 only  
 // option.text 抛出异常 (#14686, #14858)  
 // Strip and collapse whitespace  
 // https://html.spec.whatwg.org/#strip-and-collapse-whitespace  
 stripAndCollapse( jQuery.text( elem ) );  
 }  
 },  
 // 对于select的，val取值，是获取选中option对应的value值  
 select: {  
 get: function( elem ) {  
 var value, option, i,  
 options = elem.options,  
 index = elem.selectedIndex, //选中的索引，如果没有选中的话，默认为0  
 one = elem.type === "select-one",  
 values = one ? null : [], // 如是单选，values为null  
 max = one ? index + 1 : options.length;// 如是单选，max=选中的option序号+1  
 // 确定循环开始 i，如选中的第三个，index为2，故总2开始循环  
 if ( index < 0 ) {  
 i = max;  
  
 } else {  
 i = one ? index : 0;  
 }  
  
 // 循环options  
 for ( ; i < max; i++ ) {  
 option = options[ i ];  
  
 // Support: IE <=9 only  
 // IE8-9 reset后不更新selected值(#2551)  
 if ( ( option.selected || i === index ) &&  
  
 //如果option是禁用的，或者被禁用的optGroup元素中的option，不返回值  
 // <option disabled="disabled"> 或者 <optgroup disabled="disabled"><option>111</option></optgroup>  
 !option.disabled &&  
 ( !option.parentNode.disabled ||  
 !nodeName( option.parentNode, "optgroup" ) ) ) {  
  
 // 获取响应的option值  
 value = jQuery( option ).val();  
  
 // 对于单选，不需要arr  
 if ( one ) {  
 return value;  
 }  
  
 //多选返回数组  
 values.push( value );  
 }  
 }  
  
 return values;  
 },  
  
 set: function( elem, value ) {  
 var optionSet, option,  
 options = elem.options,  
 values = jQuery.makeArray( value ),  
 i = options.length;  
  
 while ( i-- ) {  
 option = options[ i ];  
  
 /\* eslint-disable no-cond-assign \*/  
 // eslint 禁止在条件语句中出现赋值操作符  
 // 在条件语句中，很容易将一个比较运算符（像 ==）错写成赋值运算符（如 =）  
 // 这里是对val可以传入数组的处理  
 // <select id="a" multiple="multiple">  
 // <option >Multiple</option>  
 // <option value="Multiple2">11</option>  
 // <option >Multiple3</option>  
 // </select>  
 // $('#a').val(['Multiple','Multiple2'])会将第一个与第二个选中，因为jQuery.valHooks.option.get( option )  
 // 获取的值分别为Multiple，Multiple2，在传入数组values里面  
 if ( option.selected =  
 jQuery.inArray( jQuery.valHooks.option.get( option ), values ) > -1  
 ) {  
 optionSet = true;  
 }  
  
 /\* eslint-enable no-cond-assign \*/  
 }  
  
 // 如无匹配的值，则将selectedIndex修正为-1  
 if ( !optionSet ) {  
 elem.selectedIndex = -1;  
 }  
 return values;  
 }  
 }  
 }  
} );

### valHooks.radios,valHooks.checkbox---------(L8089)

// Radios and checkboxes 的valHooks  
jQuery.each( [ "radio", "checkbox" ], function() {  
 jQuery.valHooks[ this ] = {  
 // 与option类似，如在数组中，则设置选中  
 set: function( elem, value ) {  
 if ( Array.isArray( value ) ) {  
 return ( elem.checked = jQuery.inArray( jQuery( elem ).val(), value ) > -1 );  
 }  
 }  
 };  
 // 对于android的bug  
 if ( !support.checkOn ) {  
 jQuery.valHooks[ this ].get = function( elem ) {  
 return elem.getAttribute( "value" ) === null ? "on" : elem.value;  
 };  
 }  
} );

### 源码解析---------------------------------(L7929)

jQuery.fn.extend( {  
 /\*\*  
 \* 获取匹配元素第一个元素的值，设置匹配集合的每个元素值  
 \* 本质是调用elem.value值获取  
 \* 所有可以拥有 value 属性的标签都是相同的，即在对一个元素调用 .val() 函数时，首先取其 value 属性的值，如果没有的话，再使用其 text 值。  
 \* 该函数常用于设置或获取表单元素的value属性值。例如：<input>、<textarea>、<select>、<option>、<button>等。  
 \* value可以传入数组：  
 \* <select id="a" multiple="multiple">  
 \* <option >Multiple</option>  
 \* <option value="Multiple2">11</option>  
 \* <option >Multiple3</option>  
 \* </select>  
 \* $('#a').val(['Multiple','Multiple2'])会将第一个与第二个选中  
 \* @param value  
 \* @return {\*}  
 \*/  
 val: function( value ) {  
 var hooks, ret, isFunction,  
 elem = this[ 0 ];  
 // 如无参数，$().val()，  
 if ( !arguments.length ) {  
 if ( elem ) {  
 // 思路与attr，prop类似  
 hooks = jQuery.valHooks[ elem.type ] ||  
 jQuery.valHooks[ elem.nodeName.toLowerCase() ];  
  
 if ( hooks &&  
 "get" in hooks &&  
 ( ret = hooks.get( elem, "value" ) ) !== undefined  
 ) {  
 return ret;  
 }  
  
 ret = elem.value;  
  
 // 处理最常见的字符串情况  
 if ( typeof ret === "string" ) {  
 return ret.replace( rreturn, "" ); // 将回车符换为""  
 }  
  
 // 如值==null，则返回""  
 return ret == null ? "" : ret;  
 }  
  
 return;  
 }  
  
 isFunction = jQuery.isFunction( value ); // 判断value是否为function  
  
 return this.each( function( i ) {  
 var val;  
 // 如非元素节点直接返回  
 if ( this.nodeType !== 1 ) {  
 return;  
 }  
  
 if ( isFunction ) {  
 // 如为函数，则传入序号i，与当前元素的值  
 val = value.call( this, i, jQuery( this ).val() );  
 } else {  
 val = value;  
 }  
  
 // 将null/undefined转换为""  
 if ( val == null ) {  
 val = "";  
 // 将数值转换为字符串形式  
 } else if ( typeof val === "number" ) {  
 val += "";  
 /// 如val值为arr，则也将值进行转换字符串形式  
 } else if ( Array.isArray( val ) ) {  
 val = jQuery.map( val, function( value ) {  
 return value == null ? "" : value + "";  
 } );  
 }  
  
 hooks = jQuery.valHooks[ this.type ] || jQuery.valHooks[ this.nodeName.toLowerCase() ];  
  
 // 如hooks的set函数返回undefined，则使用this.value形式设置值  
 if ( !hooks || !( "set" in hooks ) || hooks.set( this, val, "value" ) === undefined ) {  
 this.value = val;  
 }  
 } );  
 }  
} );