#jQuery框架

# jQuery ☞ attr&css

☞ attr & prop

**☞ 1.attr**

- 实现思路：

> 1、使用一个变量存储arguments的length属性，
>
> 2、如果length为1，name为字符串，则获取第一个元素(this[0])的对应属性节点值
>
> 3、如果length为1，name为对象，则遍历所有元素，分别给他们设置对象中所有配置的属性节点。
>
> 4、如果length为2，则遍历所有元素，分别以name为key，value为值，添加属性节点。
>
> 5、如果是设置，return this； 如果是获取，return 属性值。
>
> 补充：attr是设置或获取属性节点，所以需要通过getAttribute和setAttribute方法来实现。

- each实现方式

```
/*
* function { attr } 获取元素或者给元素设置属性节点
* param { name: string || object }
* param { value: string }
* return { 属性值 || this }
* */

attr: function (name, value) {

    var len = arguments.length;

    // 如果是null、undefined不做任何处理
    if (name == null) {
        return this;
    }

    // 如果传入一个参数
    if (len == 1) {

        // 1个参数，并且是字符串，则返回第一个元素的属性值
        if (jQuery.isString(name)) {
            return this[0] && this[0].getAttribute(name);
        }

        // 1个参数，并且是对象
        else if (typeof name == 'object') {

            // 遍历所有元素
            this.each(function () {
                var self = this;

                // 遍历所有要设置的属性key与value，
                // 备注：这里不能jQuery( name ).each，因为name不是伪数组。
                jQuery.each(name, function (key, value) {

                    // 分别给每一个元素设置所有的属性值
                    self.setAttribute(key, value);
                });
            });
        }
    }
    // 如果传入二个参数
    else if (len == 2) {
        this.each(function () {
            this.setAttribute(name, value);
        });
    }

    // 为了链式编程
    return this;
},
```

- for循环实现方式

```
/*
 * function { attr } 获取元素或者给元素设置属性节点
 * param { name: string || object }
 * param { value: string }
 * return { 属性值 || this }
 * */
_attr: function (name, value) {
    var len = arguments.length;

    // null、undefined
    if (name == null) {
        return this;
    }

    if (len == 1) {

        if (jQuery.isString(name)) {
            return this[0] && this[0].getAttribute(name);
        }

        else if (typeof name == 'object') {
            for (var i = 0, len = this.length; i < len; i++) {
                for (var key in name) {
                    this[i].setAttribute(key, name[key]);
                }
            }
        }

    }

    else if (len == 2) {
        for (var i = 0, len = this.length; i < len; i++) {
            this[i].setAttribute(name, value);
        }
    }

    // 为了链式编程
    return this;
}
```

#### ☞ 2.prop

- 实现思路：

> 1、使用一个变量存储arguments的length属性，
>
> 2、如果length为1，name为字符串，则获取第一个元素(this[0])的对应属性值
>
> 3、如果length为1，name为对象，则遍历所有元素，分别给他们设置对象中所有配置的属性。
>
> 4、如果length为2，则遍历所有元素，分别以name为key，value为值，添加属性。
>
> 5、如果是设置，return this； 如果是获取，return 属性值。
>
> 补充：prop是设置或获取属性，所以需要元素.属性名来获取，或者元素.属性名 = 属性值的方法来设置。

```
/*
 * function { prop } 获取元素或者给元素设置属性
 * param { name: string || object }
 * param { value: string }
 * return { 属性值 || this }
 * */
prop: function (name, value) {

    var len = arguments.length;

    // 如果传入1个参数
    if (len == 1) {

        // 如果是字符串
        if (jQuery.isString(name)) {
            return this[0] && this[0][name];
        }
```

```
        // 如果是对象
        else if (typeof name == 'object') {
            this.each(function () {
                var self = this;
                jQuery.each(name, function (key, value) {
                    self[key] = value;
                });
            });
        }

    }

    // 如果传入2个参数
    else if (len == 2) {
        // name作为key，value作为值，赋给每一个元素。
        this.each(function () {
            this[name] = value;
        });
    }
}
```

☞ html

- 实现思路：

> 1、如果传入undefined，不做处理，
>
> 2、如果传入null，遍历所有的元素，设置他们的innerHTML为空字符串，
>
> 3、如果没有传参，则获取第一个元素的innerHTML，
>
> 4、如果传入了html，那么遍历所有的元素，设置他们的innerHTML为传入的参数，
>
> 备注，如果是获取，则返回第一个元素的innerHTML；如果是设置，则返回this。

- 普通实现思路:

```
/*
 * function { html } 设置或获取或删除元素的内容
 * param { html: string || 没有 }
 * */
html: function (html) {

    var len = arguments.length;

    // 如果传入undefined
    if (len == 1 && html == undefined) {
        return this;
    }

    // 如果传入null,清除所有元素内容
    else if (html === null) {
        this.each(function () {
            this.innerHTML = '';
        });
    }

    // 如果没有传参，获取第一个元素内容
    else if (len == 0) {
        return this[0] && this[0].innerHTML;
    }

    // 如果传参了
    else {
        this.each(function () {
            this.innerHTML = html;
        });
    }

    // 为了链式编程
    return this;
}
```

- 借用prop方式实现html:

```
_html: function (html) {

    var len = arguments.length;

    // 如果传入undefined，嘛都不做，返回this
    if (len == 1 && html === undefined) {
        return this;
    }

    // 如果嘛都不传，借用prop方法，获取第一个元素的innerHTML，并返回prop的结果
    else if (len == 0) {
        return this.prop('innerHTML');
    }

    // 如果传入null，意指把innerHTML设置为''
    if (html === null) {
        html = '';
    }

    // 借用prop方法把所有元素的innerHTML设置为指定的值，并返回prop的结果(prop返回this)
    return this.prop('innerHTML', html);
}
```

☞ text

- 实现思路：

> 1、如果传入undefined，不做处理，
>
> 2、如果传入null，遍历所有的元素，设置他们的innerText为空字符串，
>
> 3、如果没有传参，则获取所有元素的innerText，
>
> 4、如果传入了text，那么遍历所有的元素，设置他们的innerText为传入的参数，
>
> 备注，如果是获取，则返回所有元素的innerText；如果是设置，则返回this。

```
/*
 * function { text } 获取所有元素的文本，或者清除所有元素的内容，或者给所有元素设置新的文本
 * param { text: string || 没有 || null }
 * */
text: function (text) {

    var len = arguments.length,
            result = '';

    // 传入undefined
    if (len == 1 && text === undefined) {
        return this;
    }

    // 没有传
    else if (len == 0) {
        this.each(function () {
            result += this.innerText;
        });
        return result;
    }

    // 传入null
    else if (text === null) {
        this.each(function () {
            this.innerText = '';
        });
    }

    // 传入其他内容
    else {
        this.each(function () {
            this.innerText = text;
        });
    }

    // 链式编程
    return this;
},
```

- map方法实现循环

```
  _text: function (text) {

      var len = arguments.length;

      // 传入undefined
      if (len == 1 && text === undefined) {
          return this;
      }

      // 没有传，则把所有元素的innerText拼在一起返回
      else if (len == 0) {

          /*
           * map方法用来遍历每一个元素，
           * 然后把遍历到的数据传入回调，
           * 然后把回调的结果一一保存下来，
           * 共同组成一个新数组返回，
           * 数组有一个join方法，
           * 用来把数组的内容拼接成字符串，
           * 最终通过这种方式得到了所有元素innerText拼起来的字符串。
           * */
          return this.map(function (val) {
              return val.innerText;
          }).join('');
      }

      // 传入null，意指设置text为空字符串
      if (text === null) {
          text = '';
      }

      // 统一交由prop设置innerText
      return this.prop('innerText', text);
  }
```

☞ val

- 实现思路：

> 1、如果没有传参，则返回第一个元素的value属性值( 可以考虑借用prop )
>
> 2、如果传入的是null或undefined，则设置所有元素的value属性值为''( 可以考虑借用prop )
>
> 3、如果传入其他数据，则设置所有元素的value属性值为指定的值( 可以考虑借用prop )
>
> 备注：如果没有传参，返回第一个元素的value属性值，否则返回this。

```
  /*
   * function { val } 获取第一个元素的value属性值，或者（清除||设置）所有元素的value属性值
   * param { value: 没有 || null || undefined || string }
   * */
  val: function (value) {

      var len = arguments.length;

      // 没有传参
      if (len == 0) {
          return this.prop('value');
      }

      // 如果null或者undefined，则设置的value为空字符串
      if (value == null) {
          value = ''
      }

      return this.prop('value', value);
  }
```

☞ css

- 在jQuery上添加方法

```
$.extend({
    getStyle: function (ele, styleName) {

        // 如果不传入ele，返回undefined
        if (!jQuery.isDOM(ele)) {
            return;
        }

        // 兼容获取指定的样式
        if (window.getComputedStyle) {
            return getComputedStyle(ele)[styleName];
        } else {
            return ele.currentStyle[styleName];
        }
    }
});
```

- 实现思路：

> 1、如果不传参，则返回this 2、如果传入1个参数为字符串，则返回第一个元素的指定样式 3、如果传入1个参数为对象，则给所有元素批量添加样式 4、如果传入2个参数，则给所有元素添加指定的样式

```
/*
 * function { css } 获取第一个元素的指定的样式值，或者批量设置样式值
 * param { value: 没有 || null || undefined || string }
 * */
css: function (style, value) {

    var len = arguments.length;

    // 如果不传参，则返回this
    if (len == 0) {
        return this;
    }

    if (len == 1) {

        // 如果传入1个参数为字符串，则返回第一个元素的指定样式
        if (jQuery.isString(style)) {
            return jQuery.getStyle(this[0], style);
        }

        // 如果传入1个参数为对象，则给所有元素批量添加样式
        else if (typeof style == 'object') {
            this.each(function () {
                var self = this;
                jQuery.each(style, function (key, val) {
                    self.style[key] = val;
                });
            });
        }

    }

    // 如果传入2个参数，则给所有元素添加指定的样式
    else if (len == 2) {
        this.each(function () {
            this.style[style] = value;
        });
    }

    // 为了链式编程
    return this;
}
```