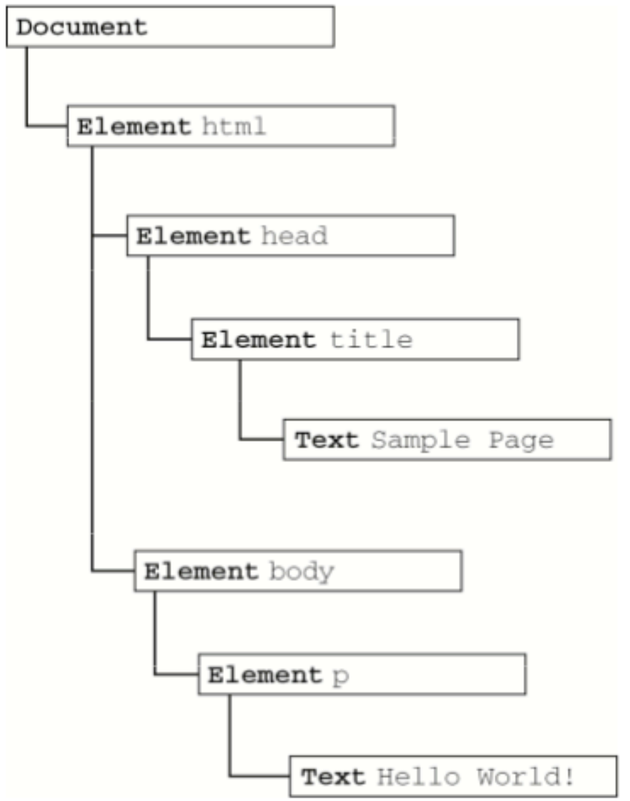
DOM编程

DOM基础
Document节点
DOM元素节点
DOM操作
DOM查找
DOM级别（动态样式）
元素大小

DOM基础

DOM基础 – 节点层级

```
<html>
<head>
    <title>Sample Page</title>
</head>
<body>
    <p>Hello World!</p>
</body>
</html>
```



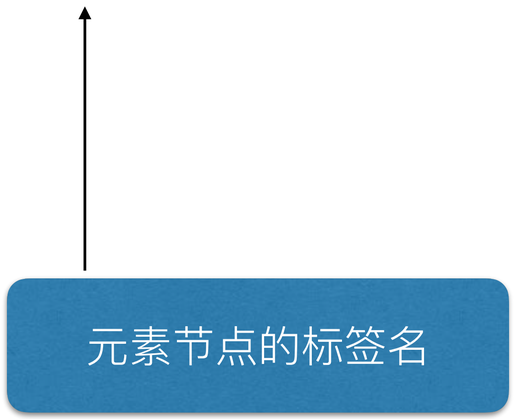
Document节点
元素节点
文本节点

DOM基础 – 节点类型

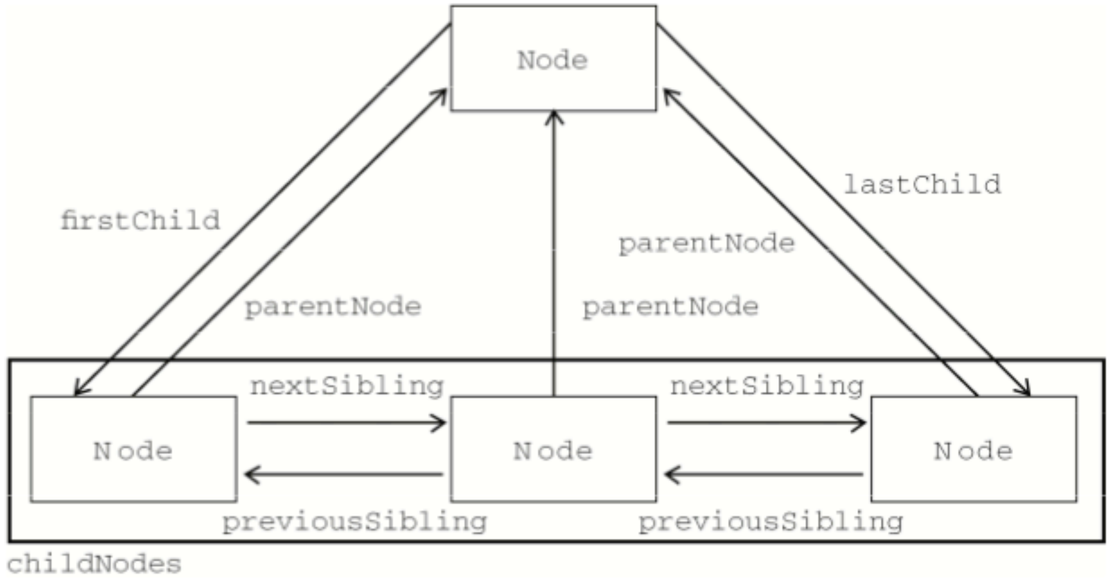
Node.ELEMENT_NODE(1)
Node.ATTRIBUTE_NODE(2)
Node.TEXT_NODE(3)
Node.CDATA_SECTION_NODE(4)
Node.ENTITY_REFERENCE_NODE(5)
Node.ENTITY_NODE(6)
Node.PROCESSING_INSTRUCTION_NODE(7)
Node.COMMENT_NODE(8)
Node.DOCUMENT_NODE(9)
Node.DOCUMENT_TYPE_NODE(10)
Node.DOCUMENT_FRAGMENT_NODE(11)
Node.NOTATION_NODE(12)

DOM基础 – 判断元素节点

```
if (someNode.nodeType == 1){
    value = someNode.nodeName;
}
```



DOM基础 – 节点关系



NodeList

父子关系

兄弟关系

DOM基础 – NodeList

访问NodeList中的节点

someNode.childNodes[0]
someNode.childNodes.item(1)

```
var arrayOfNodes = Array.prototype.slice.call(someNode.childNodes,0);
```

```
function toArrArgs() {

    var args = Array.protoype.slice.call(arguments, 0);

}
```

Document节点

Document节点 – 特征

- nodeType 的值为 9;
- nodeName 的值为"#document";
- nodeValue 的值为 null;
- parentNode 的值为 null;
- ownerDocument 的值为 null;
- 其子节点可能是一个 DocumentType(最多一个)、Element(最多一个)、ProcessingInstruction 或 Comment。

Document节点

```
<html>
    <body>

    </body>
</html>
```

```
var html = document.documentElement;
alert(html === document.childNodes[0]);
alert(html === document.firstChild);
```

Document节点 – 属性

document.title

document.referrer

document.domain

DOM元素节点

DOM元素节点 – 默认属性

```
<div id="someDiv" class="cls" title="text" lang="en" dir="ltr"></div>

element.id //someDiv
element.className //cls
element.title //text
element.lang //en
element.dir //ltr
```

DOM元素节点 – 属性

```
<div id="someDiv" class="cls" title="text" lang="en" dir="ltr" myattr="val"></div>

element.getAttribute("id")
element.getAttribute("class")
element.getAttribute("title")
element.getAttribute("lang")
element.getAttribute("dir")
element.getAttribute("val")
```

getAttribute

setAttribute

removeAttribute

DOM操作

DOM操作 – 创建

```
document.createElement(“div”)
```

DOM操作 – 添加

添加新节点

```
var returnedNode = someNode.appendChild(newNode);
alert(returnedNode == newNode);          //true
alert(someNode.lastChild == newNode);    //true
```

添加已有节点

```
//someNode 有多个子节点
var returnedNode = someNode.appendChild(someNode.firstChild);
alert(returnedNode == someNode.firstChild);      //false
alert(returnedNode == someNode.lastChild);       //true
```

DOM操作 – 插入

```
//插入后成为最后一个子节点
returnedNode = someNode.insertBefore(newNode, null);
alert(newNode == someNode.lastChild);   //true
```

```
//插入后成为第一个子节点
var returnedNode = someNode.insertBefore(newNode, someNode.firstChild);
alert(returnedNode == newNode);         //true
alert(newNode == someNode.firstChild);  //true
```

```
//插入到最后一个子节点前面
returnedNode = someNode.insertBefore(newNode, someNode.lastChild);
alert(newNode == someNode.childNodes[someNode.childNodes.length-2]); //true
```

DOM操作 – 替换

```
//替换第一个子节点
var returnedNode = someNode.replaceChild(newNode, someNode.firstChild);

//替换最后一个子节点
returnedNode = someNode.replaceChild(newNode, someNode.lastChild);
```

DOM操作 – 删除

```
//移除第一个子节点
var formerFirstChild = someNode.removeChild(someNode.firstChild);

//移除最后一个子节点
var formerLastChild = someNode.removeChild(someNode.lastChild);
```

DOM操作 – 克隆

```
<ul>
    <li>item 1</li>
    <li>item 2</li>
    <li>item 3</li>
</ul>
```

```
var deepList = myList.cloneNode(true);
```

```
var shallowList = myList.cloneNode(false);
```

DOM操作 – 效率

innerHTML

outerHTML

```
<ul>
    <li>List item 0</li>
    <li>List item 1</li>
    <li>List item 2</li>
    ...
    <li>List item 9</li>
</ul>
```

```
for(var x = 0; x < 10; x++) {
    var li = document.createElement("li");
    li.innerHTML = "List item " + x;
    listNode.appendChild(li);
}
```

```
var html = "";
for(var x = 0; x < 10; x++) {
    html += "<li>List item " + x + "</li>";
}
listNode.innerHTML = html;
```

DOM操作 – 效率

DocumentFragment

```
<ul>
    <li>List item 0</li>
    <li>List item 1</li>
    <li>List item 2</li>
    ...
    <li>List item 9</li>
</ul>
```

```
for(var x = 0; x < 10; x++) {
    var li = document.createElement("li");
    li.innerHTML = "List item " + x;
    listNode.appendChild(li);
}
```

```
var frag = document.createDocumentFragment();
for(var x = 0; x < 10; x++) {
    var li = document.createElement("li");
    li.innerHTML = "List item " + x;
    frag.appendChild(li);
}
listNode.appendChild(frag);
```

DOM操作 – innerHTML vs DocumentFragment

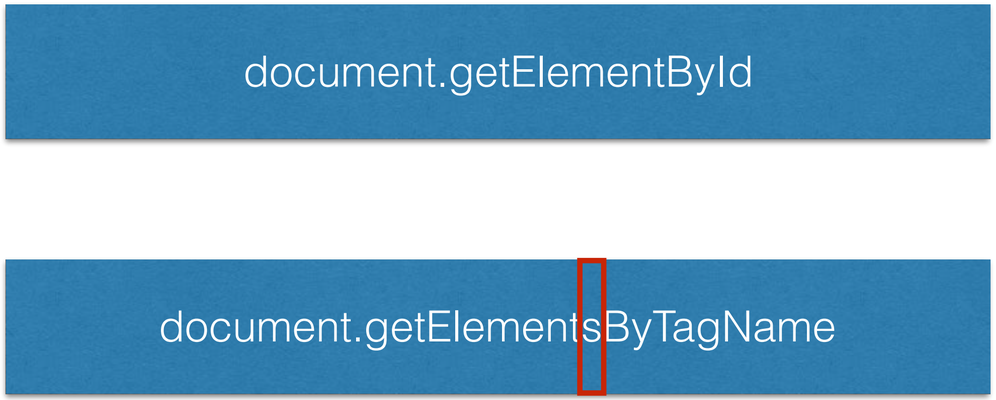
```
<ul>
    <li>List item -1</li>
    <li>List item 0</li>
    <li>List item 1</li>
    <li>List item 2</li>
    ...
    <li>List item 9</li>
</ul>
```

```
listNode.innerHTML += html;
```

```
listNode.appendChild(frag);
```

DOM查找

DOM查找 – 最普通的方法



DOM查找 – 特殊集合

document.anchors

document.forms

document.images

document.links

DOM查找 – querySelector

```
//取得 body 元素
var body = document.querySelector("body");

//取得 ID 为"myDiv"的元素
var myDiv = document.querySelector("#myDiv");

//取得类为"selected"的第一个元素
var selected = document.querySelector(".selected");

//取得类为"button"的第一个图像元素
var img = document.body.querySelector("img.button");
```

DOM查找 – querySelectorAll

```
//取得某<div>中的所有<em>元素（类似于 getElementsByTagName("em")）
var ems = document.getElementById("myDiv").querySelectorAll("em");

//取得类为"selected"的所有元素
var selecteds = document.querySelectorAll(".selected");

//取得所有<p>元素中的所有<strong>元素
var strongs = document.querySelectorAll("p strong");
```

DOM级别（动态样式）

DOM1
DOM2
DOM3

样式处理

CSS属性	JavaScript属性
background-image	style.backgroundImage
color	style.color
display	style.display
font-family	style.fontFamily

```
var myDiv = document.getElementById("myDiv");

//设置背景颜色
myDiv.style.backgroundColor = "red";

//改变大小
myDiv.style.width = "100px";
myDiv.style.height = "200px";

//指定边框
myDiv.style.border = "1px solid black";
```

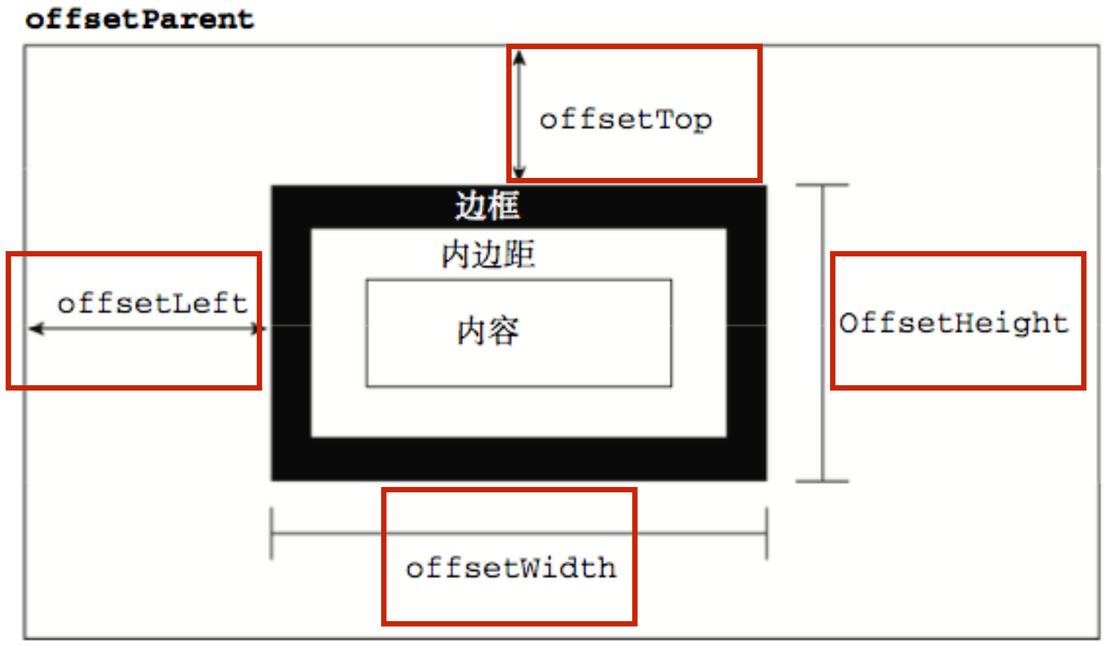
计算样式

```
<!DOCTYPE html>
<html>
<head>
    <title>Computed Styles Example</title>
    <style type="text/css">
        #myDiv {
            background-color: blue;
            width: 100px;
            height: 200px;
        }
    </style>
</head>
<body>
    <div id="myDiv" style="background-color: red; border: 1px solid black"></div>
</body>
</html>
```

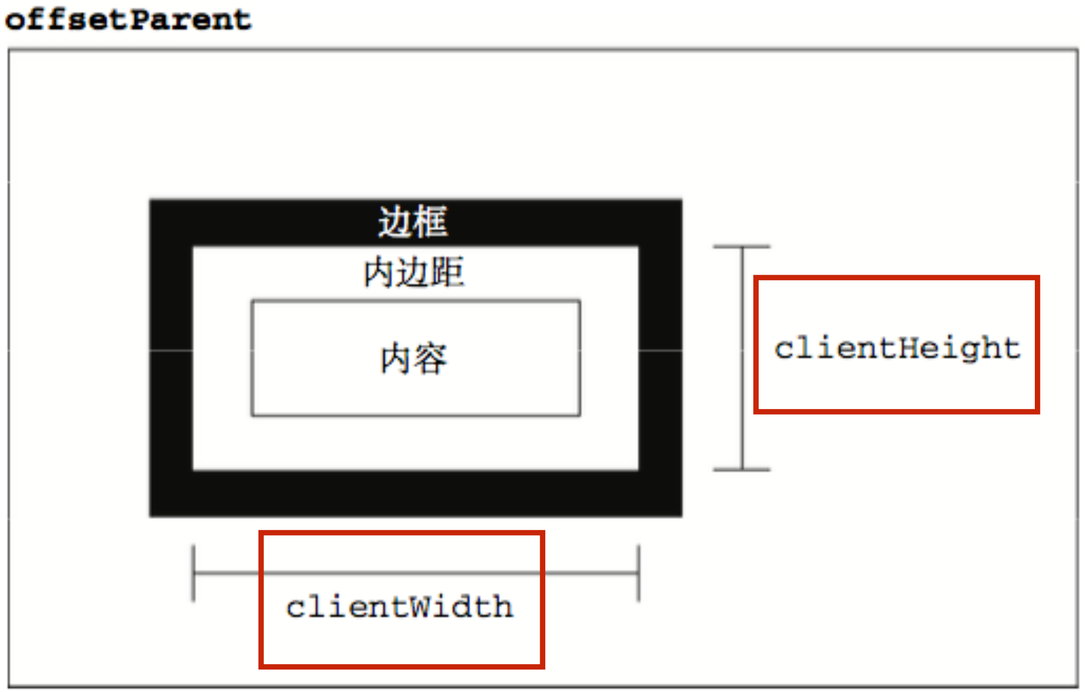
```
var myDiv = document.getElementById("myDiv");
var computedStyle = document.defaultView.getComputedStyle(myDiv, null);

alert(computedStyle.backgroundColor);   // "red"
alert(computedStyle.width);             // "100px"
alert(computedStyle.height);            // "200px"
alert(computedStyle.border);            // 在某些浏览器中是"1px solid black"
```

元素大小

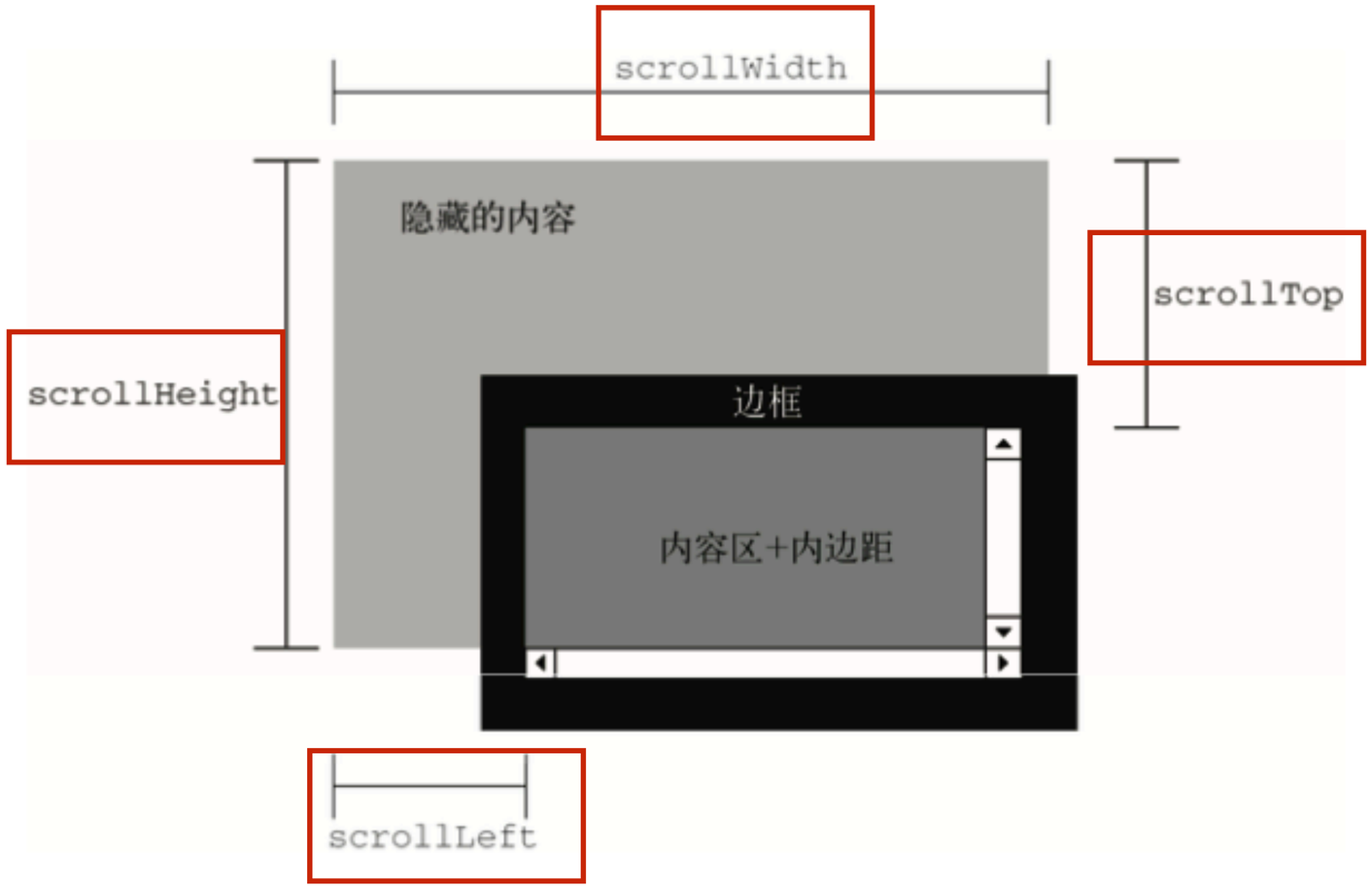
元素大小 – 偏移


offsetParent
offsetTop
边框
内边距
offsetLeft
内容
OffsetHeight
offsetWidth

元素大小 – 客户区



滚动大小 – 滚动

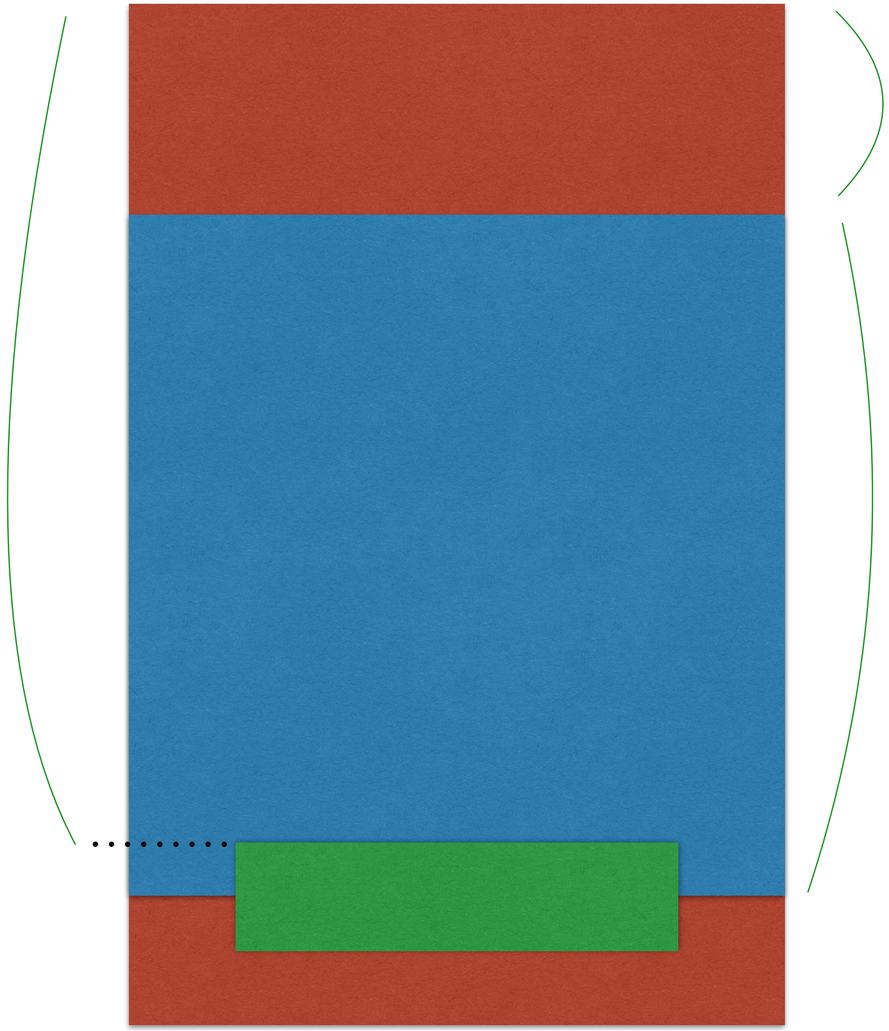

scrollWidth
隐藏的内容
scrollHeight
scrollTop
边框
内容区+内边距
scrollLeft

getBoundingClientRect

```
{
    left,
    right,
    top,
    bottom
}
```

图片懒加载问题

```
<div id="div1">
    <ul>
        <li><img src="http://xxx/7620cd6e740c3c50.jpg" /></li>
        <li><img src="http://xxx/2c67d8e270eb61c3.jpg" /></li>
        <li><img src="http://xxx/9c9d058c0c731087.jpg" /></li>
        <li><img src="http://xxx/007da8d5456bc0ab.jpg" /></li>
        <li><img src="http://xxx/64d939a5456bc0ab.jpg" /></li>
        <li><img src="http://xxx/95eb5da5456bc0ab.jpg" /></li>
        <li><img src="http://xxx/5c3fa2b5456bc0ab.jpg" /></li>
        <li><img src="http://xxx/f6443925456bc0ab.jpg" /></li>
        <li><img src="http://xxx/f96780b5456bc0ab.jpg" /></li>
        <li><img src="http://xxx/54a75d2e740c3c50.jpg" /></li>
        <li><img src="http://xxx/3b173ee50fdad689.jpg" /></li>
        <li><img src="http://xxx/95eb5da5456bc0ab.jpg" /></li>
        <li><img src="http://xxx/5c3fa2b5456bc0ab.jpg" /></li>
        <li><img src="http://xxx/f6443925456bc0ab.jpg" /></li>
        <li><img src="http://xxx/f96780b5456bc0ab.jpg" /></li>
        <li><img src="http://xxx/54a75d2e740c3c50.jpg" /></li>
        <li><img src="http://xxx/3b173ee50fdad689.jpg" /></li>
    </ul>
</div>
```



页面的scrollTop

元素的offsetTop

页面的clientHeight

元素的offsetTop < 页面的scrollTop + 页面的clientHeight

src -> data-src

```
<div id="div1">
    <ul>
        <li data-src="http://xxx/7620cd6e740c3c50.jpg"></li>
        <li data-src="http://xxx/2c67d8e270eb61c3.jpg"></li>
        <li data-src="http://xxx/9c9d058c0c731087.jpg"></li>
        <li data-src="http://xxx/007da8d5456bc0ab.jpg"></li>
        <li data-src="http://xxx/64d939a5456bc0ab.jpg"></li>
        <li data-src="http://xxx/95eb5da5456bc0ab.jpg"></li>
        <li data-src="http://xxx/5c3fa2b5456bc0ab.jpg"></li>
        <li data-src="http://xxx/f6443925456bc0ab.jpg"></li>
        <li data-src="http://xxx/f96780b5456bc0ab.jpg"></li>
        <li data-src="http://xxx/54a75d2e740c3c50.jpg"></li>
        <li data-src="http://xxx/3b173ee50fdad689.jpg"></li>
        <li data-src="http://xxx/95eb5da5456bc0ab.jpg"></li>
        <li data-src="http://xxx/5c3fa2b5456bc0ab.jpg"></li>
        <li data-src="http://xxx/f6443925456bc0ab.jpg"></li>
        <li data-src="http://xxx/f96780b5456bc0ab.jpg"></li>
        <li data-src="http://xxx/54a75d2e740c3c50.jpg"></li>
        <li data-src="http://xxx/3b173ee50fdad689.jpg"></li>
    </ul>
</div>
```

通过data-src加载图片

```
function setImg(index) {

    var aLi = document.getElementsByName("li");

    if (aLi[index].childNodes.length == 0) {
        if (aLi[index].dataset) {
            var src = aLi[index].dataset.src;
        } else {
            var src = aLi[index].getAttribute('data-src');
        }
        var oImg = document.createElement('img');
        oImg.src = src;
        aLi[index].appendChild(oImg);
    }
}
```

获取元素距离页面顶部的距离

```
function getH(el) {
    var h = 0;
    while (el) {
        h += el.offsetTop;
        el = el.offsetParent;
    }
    return h;
}
```

组装

```
window.onscroll = function () {

    var aLi = document.getElementsByName("li");

    for (var i = 0, l = aLi.length; i < l; i++) {
        var oLi = aLi[i];
        var t = document.documentElement.clientHeight + (document.documentElement.scrollTop || document.body.scrollTop);
        var h = getH(oLi);
        if (h < t) {
            setTimeout("setImg(" + i + ")", 500);

            //todo, 优化
        }
    }
};
```

```
window.onload = function () {
    window.onscroll();
};
```