- DOM
  - document object model 文档对象模型，将网页转换为dom对象，因此脚本可以curd dom
  - dom 是一个接口规范，可以用任意种语言实现dom 结构，所以dom 并不是JavaScript 的一部分，但是 DOM 操作是 JavaScript 最常见的任务，离开了 DOM，JavaScript 就无法控制网页。
  - node ,DOM中最小单位
    - 文档的树形结构（ DOM 树），就是由各种不同类型的节点组成。每个节点可以看作是文档树的一片叶子。
    - 7种节点类型1. Document 整个文档结构的顶层节点2. DocumentType doctype 标签3. Element 网页中的各种标签 例如 <body> t; <a>4. Attribute 网页元素的各种属性 class= 'red'5. Text 标签之间或者标签包含的文本6. Comment 注释7. Document Fragment 文档的片段
    - 以上的七种节点都继承了Node接口的属性和方法这是DOM操作的基础
      - Node.nodeType 返回一个数值类型，表示节点类型文档类型document：9元素节点element：1属性节点attribute：2文本节点 text：3文档片断节点（DocumentFragment）：11，对应常量Node.DOCUMENT_FRAGMENT_NODE文档类型节点（DocumentType）：10，对应常量Node.DOCUMENT_TYPE_NODE注释节点（Comment）：8，对应常量Node.COMMENT_NODE
      - Node.nodeName节点名称文档节点（document）：#document元素节点（element）：大写的标签名属性节点（attr）：属性的名称文本节点（text）：#text文档片断节点（DocumentFragment）：#document-fragment文档类型节点（DocumentType）：文档的类型注释节点（Comment）：#comment
      - Node.nodeValue 节点的字符串值，表示当前节点本身的值，可读写只有文本节点（text）和注释节点（comment）有文本值，因此这两类节点的nodeValue可以返回结果，其他类型的节点一律返回null。
      - Node.textContent返回当前节点和它的所有后代节点的文本内容textContent属性自动忽略当前节点内部的 HTML 标签，返回所有文本内容。该属性是可读写的，设置该属性的值，会用一个新的文本节点，替换所有原来的子节点。已经有一个好处，就是自动对 HTML 标签转义。这很适合用于用户提供的内容
      - Node.baseURI返回一个字符串，表示当前网页的绝对路径。浏览器根据这个属性，计算网页上的相对路径的 URL。该属性为只读。
      - Node.nextSibling属性返回紧最在当前节点后面的第一个同级节点。如果当前节点后面没有同级节点，则返回null。
      - Node.previousSibling属性返回当前节点前面的、距离最近的一个同级节点。如果当前节点前面没有同级节点，则返回null。
      - Node.parentNode属性返回当前节点的父节点。对于一个节点来说，它的父节点只可能是三种类型：元素节点（element）、文档节点（document）和文档片段节点（documentfragment）
      - Node.firstChild属性返回当前节点的第一个子节点，如果当前节点没有子节点，则返回null。firstChild返回的除了元素节点，还可能是文本节点或注释节点。Node.lastChild属性返回当前节点的最后一个子节点，如果当前节点没有子节点，则返回null。用法与firstChild属性相同。
      - Node.childNodes属性返回一个类似数组的对象（NodeList集合），成员包括当前节点的所有子节点document.querySelectorAll's.childNodes
    - Node接口的方法
      - Node.appendChild() 接收一个节点作为参数，当做最后的子节点插入当前节点。该方法的返回值就是插入文档的子节点。
      - Node.hasChildNodes() 返回一个布尔值，表示当前节点是否有子节点
      - Node.cloneNode() 方法用于克隆一个节点，接收一个布尔值作为参数，表示是否克隆子节点，返回新克隆的节点，也会拷贝节点的所有属性，但是会失去 addEventListner方法和on-属性
      - Node.insertBefore()用于将某个节点插入父节点内部的指定位置。parentNode.insertBefore(newNode, referenceNode)insertBefore方法接受两个参数，第一个参数是所要插入的节点newNode，第二个参数是父节点parentNode内部的一个子节点referenceNode。newNode将插在referenceNode这个子节点的前面。返回值是插入的新节点newNode。
      - Node.removeChild()接受一个子节点作为参数，用于从当前节点移除该子节点。返回值是移除的子节点。
      - Node.replaceChild()方法用于将一个新的节点，替换当前节点的某一个子节点。parentNode.replaceChild(newChild, oldChild)replaceChild方法接受两个参数，第一个参数newChild是用来替换的新节点，第二个参数oldChild是将要替换走的子节点。返回值是替换走的那个节点oldChild
    - NodeList 接口
      - 一个类似数组的对象，成员为节点对象Node.childNodesdocument.querySelectorAll(), document.getElementsByTagName()等节点方法NodeList 类数组中只有Node.childNodes返回的是动态的，节点的删除和添加，替换，都会立刻在Node.childNodes中体现document.querySelectorAll() 和 document.getElementsByTagName() 等方法都是静态的返回一个类数组
      - NodeList.prototype.length, NodeList.prototype.forEach() 都是NodeList类数组的方法
      - NodeList.prototype.entries(), NodeList.prototype.keys(), NodeList.prototype.values() es6中的方法可以使用for of 遍历
      - NodeList.prototype.item(index) 方法返回一个成员
    - HTMLCollection 接口
      - HTMLCollection 是一个节点对象的集合，这个集合是动态的，会实时反映 DOM 的任何变化。只包含元素节点Element，只返回一个类数组对象，但是这个HTMLCollection 返回的类数组对象没有forEach() 方法
        - 返回HTMLCollection 实例的主要是Document对象集合的属性 document.links , document.forms, document.images
    - ParentNode 接口
      - 节点对象除了会继承Node接口以外，还会继承ParentNode接口，表示当前节点的是一个父节点，提供一些处理子节点的方法如果当前节点是父节点则会继承ParentNode接口，只有元素节点和Element, 文档节点Document, 文档片段节点document.createDocumentFragment 有子节点，因此这三个节点自动继承了ParentNode 接口
      - ParentNode.children 返回一个HTMLCollection 实例，返回的是当前节点的所有子节点
      - ParentNode.firstElementChild,返回当前节点的第一个元素子节点。如果没有任何元素子节点，则返回null。ParentNode.lastElementChild,返回当前节点的最后一个元素子节点，如果不存在任何元素子节点，则返回null。ParentNode.childElementCount, 返回当前节点的[子节点个数，没有子节点就返回0
      - ParentNode.append() , ParentNode.prepend() 方法ParentNode.append()附加，方法为当前节点追加一个或多个子节点，位置是最后一个元素子节点的后面。ParentNode.prepend()前置 ，方法为当前节点追加一个或多个子节点，位置是第一个元素子节点的前面。它的用法与append方法完全一致，也是没有返回值。
    - ChildNode 接口
      - 如果一个节点有父节点，那么该节点就继承了ChildNode接口。
      - ChildNode.remove() 从父级节点删除当前节点 ，el.remove()
      - ChildNode.before(), ChildNode.after() ChildNode.before()方法用于在当前节点的前面，插入一个或多个同级节点。两者拥有相同的父节点
      - ChildNode.replaceWith()使用参数节点，替换当前节点。参数可以是元素节点，也可以是文本节点
  - document对象
    - 文档对象模型第一个节点就是html标签，它构成了树结构的根节点（ root node ），其他 HTML 标签节点都是它的下级节点除了根节点，其他节点都有三层关系，父节点关系 parentNode 直接的上级节点。兄弟节点关系 sibling 有同一个父节点的节点 nextSibling 和 previousSibling 子节点关系 childNodes 直接的下级节点 firstChild 和 lastChild
    - document对象的属性
      - document对象是文档的根节点，每张网页都有自己的document 对象，window.document就是document
      - document继承了document.EventTarget 接口，Node接口，ParentNode 接口，document可以调用这些接口的属性和方法
      - document.defaultView 返回的是document的window对象
      - document.doctype指向节点，即文档类型
      - document.documentElement 属性返回当前文档的根节点 指的是html 元素节点
      - document.body指向body节点和document.head指向的head节点
      - document.activeElement属性返回获得当前焦点（focus ）的 DOM 元素
      - document.links, document.forms, document.images ， document.links 属性返回的是当前文档设置了属性为href 的所有的a, area 节点document.images 属性返回的是当前文档所有的img 节点document.forms 属性返回的是当前文档的所有forms 节点
      - document.documentURI 和document.URL 属性都是返回当前文档的网址document.domain 返回当前文档的域名，不包含协议和端口
      - document.lastModified属性返回一个字符串，表示当前文档最后修改的时间。
      - document.compatMode 返回浏览器处理文档的模式，可能的值为BackCompat（向后兼容模式）和CSS1Compat（严格模式）。
      - document.readyState 返回文档的状态，有三个值：loading：加载 HTML 代码时（尚未完成解析）interactive：加载外部资源时完成complete：加载完成
      - document.open() 和document.close()document.open方法清除当前文档所有内容，使得文档处于可写状态，供document.write方法写入内容。document.close方法用来关闭document.open打开的文档。
      - document.write() ,document.writeln() 用于向当前文档写入内容，这里编写的文本会当做HTML编译，不会转义
      - document.querySelector(), document .SelectorAll() 方法document.querySelector 接受一个 CSS 选择器作为参数，返回匹配该选择器的元素节点。如果有多个节点满足匹配条件，则返回第一个匹配的节点。如果没有发现匹配的节点，则返回null。document.querySelectorAll方法与querySelector用法类似，区别是返回一个NodeList对象，包含所有匹配给定选择器的节点。document.querySelectorAll(' .myclass')
      - document.getElementsByTagName()搜索 HTML 标签名，返回符合条件的元素。它的返回值是一个类似数组对象（HTMLCollection实例），可以实时反映 HTML 文档的变化。如果没有任何节点的元素，就返回一个空集。
      - document.getElementsByClassName方法返回一个类似数组的对象（HTMLCollection实例），包括了所有class名字符合指定条件的元素，元素的变化实时反映在返回结果中。document.getElementsByClassName('foo bar')
      - document.getElementsByName方法用于选择拥有name属性的 HTML 元素,返回一个类似数组的对象（ NodeList 实例），因为name属性相同的元素可能不止一个。
      - document.getElementById方法返回匹配指定id属性的元素节点。如果没有发现匹配的节点，则返回null。document.getElementById方法与document.querySelector方法都接受匹配元素节点，不同之处是document.querySelector方法的参数使用 CSS 选择器表达法，document.getElementById方法的参数是元素的id属性。document.getElementById('myElement')document.querySelector('#myElement'';但是document.getElementById比document.querySelector效率高得多。
      - document.createElement() 方法用来生成节点，并返回该节点，let div = document.createElement('div')并且，document.createElement("custom") 可以通自定义的标签名
      - document.createTextNode()用来生成文本节点（Text实例）.let div = document.createElement("div");let text = document.createTextNode("hello world");div.appendChild(text)这种方法可以确保返回的节点，被浏览器当作文本渲染，而不是当作 HTML 代码渲染。因此，可以用来展示用户的输入，避免 XSS 攻击。
      - document.createAttribute()方法生成一个新的属性节点（Attr实例），并返回它
      - document.createDocumentFragment()生成一个空的文档片段对象（DocumentFragment 实例），let docFragment = document.createDocumentFragment()documentFragment存在于内存中，不属于document 文档，可以用来创建一段比较复杂的DOM结构，因为documentFragment 不属于当前文档，因此修改他并不会影响当前文档的重新渲染，因此比直接修改DOM的效率更高
      - document.createEvent()方法生成一个事件对象，该对象可以用 element.dispatchEvent()触发指定事件
      - document.addEventListener() 和 document.removeEventListener()document.addEventListener("eventName", function() {}, false/true);document.removeEventListener("eventName", function(){}, false/true)document.dispatchEvent() //触发事件
    - element对象
      - Element对象对应网页的HTML元素，每一个 HTML 元素，在 DOM 树上都会转化成一个Element 节点对象（以下简称元素节点）元素节点的nodeType属性都是1,Element 对象继承了Node接口，因此Node的属性和方法在Element 对象都存在
      - 元素节点的属性
        - Element.id, 元素的id可读写
        - Element.tagName ，返回元素的HTML标签，与nodeName属性的值相等。
        - Element.dir 当前元素的文字方向
        - Element.accessKey用于读写分配给当前元素的快捷键。
        - Element.draggable属性返回一个布尔值，表示当前元素是否可拖动。该属性可读写
        - Element.title属性用来读写当前元素的 HTML 属性title。该属性通常用来指定，鼠标悬浮时弹出的文字提示框。
        - Element.attributes属性返回一个类似数组的对象，成员是当前元素节点的所有属性节点 。
        - Element.className , Element.classList.className属性用来读写当前元素节点的class属性。它的值是一个字符串，每个class之间用空格分割。classList属性返回一个类似数组的对象，当前元素节点的每个class就是这个对象的一个成员。
        - Element.dataset 属性元素可以自定义data- 属性，用来添加数据<div data-timestamp="1522907809292">
        - Element.innerHTML属性返回一个字符串，等同于该元素包含的所有 HTML 代码。该属性可读写，常用来设置某个节点的内容。它能改写所有元素节点的内容，包括和元素。如果将innerHTML属性设为空，等于删除所有它包含的所有节点。
        - Element.outerHTML属性返回一个字符串，表示当前元素节点的所有 HTML 代码，包括该元素本身和所有子元素。outerHTML属性是可读写的，对已进行赋值，等于替换掉当前元素
        - Element.clientHeight , Element.clientWidth属性返回一个整数值，表示元素节点的css 高度，只对块级元素生效，对于行内元素返回0。如果块级元素没有设置 CSS 高度，则返回实际高度。除了元素本身的高度，它还包括padding部分，/ 和口高度document.documentElement.clientHeight;/ 网页总高度document.body.clientHeight
        - Element.clientLeft , Element.clientTop属性等于元素节点左边框（left border）的宽度（单位像素），不包括左侧的paddingl和margin。如果没有设置左边框，或者图片内元素（display: inline）。该属性总是返回整数值，如果是小数，会四舍五入。Element.clientTop属性等于网页元素顶部边框的宽度（单位像素），其他特点都与clientTop相同。
        - Element.scrollHeight, Element.scrollWidth, 属性返回一个整数值，表示当前元素的总高度，包括padding，margin, border,这两个属性只读
        - Element.scrollLeft 属性表示当前元素的水平滚动条向右侧滚动的像素数量，Element.scrollTop属性表示当前元素的垂直滚动条向下滚动的像素数量。对于那些没有滚动条的网页元素，这两个属性总是等于0。这两个属性都可读写，设置该属性的值，会导致浏览器将当前元素自动滚动到相应的位置。
        - Element.offsetHeight , Element.offsetWidthElement.offsetHeight 属性返回一个整数，表示元素的 CSS 垂直高度（单位像素），包括元素本身的高度、padding 和 border，以及水平滚动条的高度（如果存在滚动条）。这两个属性都是只读属性
      - 元素节点的方法
        - 分支主题
        - 分支主题
        - 分支主题
        - 分支主题
        - 分支主题
        - 分支主题
        - 分支主题
        - 分支主题
    - event事件
      - MouseEvent 接口，鼠标事件let event = new MouseEvent(type, options)
        - mousemove鼠标在节点内部移动时触发，持续移动时继续触发
        - mouseenter鼠标进入一个节点时触发，进入子节点不会触发
        - mouseover鼠标进入时触发，进入子节点在触发一次
        - mouseout 鼠标离开时触发，鼠标离开一个节点时触发，离开父节点也会触发这个事件
        - mouseleave 鼠标离开时触发，鼠标离开一个节点时触发，离开父节点不会触发这个事件
        - mousedown鼠标按下时触发
        - mouseup鼠标释放按下时触发
        - click事件指的是，用户在同一个位置先完成mousedown动作，再完成mouseup动作。因此，触发顺序是，mousedown首先触发，mouseup接着触发，click最后触发。
        - dblclick事件前会在mousedown、mouseup、click之后触发。
        - contextmenu：按下鼠标右键时（上下文菜单出现前）触发，或者按下“上下文菜单键”时触发。
      - WheelEvent接口继承了 MouseEvent接口,鼠标滚动事件let wheel = new WheelEvent(type, options)
        - wheel 事件，鼠标滚动事件
      - KeyboardEvent接口，键盘事件let keyboard = new KeyboardEvent(type, options);
        - keydown事件，按下键盘时触发
        - keypress事件，按下有值的键时触发，对于有值的键触发顺序为 ，keydown -> keypres
        - keyup事件，松开按键时触发
      - http://javascript.ruanyifeng.com/dom/event-type.html#toc0 更多事件参考
    - event模型
      - EventTarget接口()
        - addEventListener(), removeEventListener(), dispatchEvent()
        - EventTarget.addEventListener(type, listener, useCapture)布尔值，表示监听函数是否在捕获阶段（capture）触发默认为false（监听函数只在冒泡阶段被触发），该参数可选。
      - 事件的传播
        - 一个事件触发，会在子元素和父元素之间传播，传播分为三个阶段
        - 第一个阶段，从window对象传导到目标节点（上层传到底层），称为“捕获阶段” capture
        - 第二个阶段，在目标节点上触发，称为“目标阶段” (target phase)。
        - 第三个阶段，从目标节点传导回window对象（从底层传回上层），称为“冒泡阶段” bubbling
        - event.stopPropagation() 阻止事件传播，即阻止事件的冒泡bubblingevent.stopImmediatePropagation() 阻底阻止事件的冒泡行为
      - let event = new Event(type, options)
        - event实例对象的属性有，bubbles 表示是否冒泡，默认falseevent.preventDefault() 消灭跟对该事件的默认行为
        - event.cancelable属性返回一个布尔值，表示事件是否可以取消。该属性为只读属性，一般用来了解 Event 实例的特性。
        - Event.currentTarget属性返回事件当前所在的节点，即正在执行的监听函数所绑定的那个节点。Event.target属性返回原始触发事件的那个节点，即事件最初发生的节点。事件传播过程中，不同节点的监听函数内部的Event.target与Event.currentTarget属性的值是不一样的，前者总是不变的，后者则是指向监听函数所在的那个节点对象。
    - css操作
      - 操作css最简单的方式就是element.getAttribute(), setAttribute(), removeAttribute() 读写网页元素的style 属性element.setAttribute('style', 'color: red;font-size;16px')
      - cssStyleDeclaration 接口三个地方部署了cssStyleDeclaration接口元素节点的style属性（Element.style ）CSSStyle实例的style属性window.getComputedStyle()的返回结果
        - 元素节点可以读写style 属性，css属性需要驼峰式写法. 但是这种写法只能获取取到行内样式，并不是所有的样式都能获取到let divStyle = document.querySelector('#id').style
        - divStyle.cssText可以读写所有样式文本divStyle.cssText = "color:red;"
        - divStyle.getPropertyValue("margin") //10px.接受 CSS 样式属性名作为参数，返回一个字符串，表示该属性的属性值。
        - divStyle.removeProperty()方法接受一个属性名作为参数，在 CSS 规则里面移除这个属性，返回这个属性原来的值。
        - divStyle.setProperty('styleName', 'value')方法用来设置新的 CSS 属性。该方法没有返回值。
        - window.getComputedStyle()