# 1.DOM树

|  |
| --- |
|  |

# 2.遍历DOM树,当你在console中输入document,会输出整个html文档,document的第一个孩子是html标签,以此类推

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

# 3.修改h1标签的文本

|  |  |
| --- | --- |
|  |  |
|  |  |

## 还可以单位到一个元素并且用代码来实现点击功能

|  |
| --- |
|  |

# 4.如果需要在html文件在使用js,最好把它放在body标签的所有子元素之后,否则会出现找不到元素的错误

|  |
| --- |
|  |
|  |

# 5.获取元素的几种方式

## 1)document.getElementsByTagName,获取的是一个数组

|  |
| --- |
|  |

## 2) document.getElementsByClassName,获取的是一个数组,即使只有一个元素

|  |
| --- |
|  |
|  |

## 3)document.getElementById,获取的是一个元素

|  |
| --- |
|  |
|  |

## 4)document.querySelector,需要用.classname,#idname或者是标签名称的写法,只会获取符合条件的第一个元素

|  |
| --- |
|  |
|  |
|  |
|  |

### 注意:querySelector方法里面可以使用组合选择器

|  |
| --- |
|  |
|  |

## 5)document.querySelectorAll方法,获取所有符合条件的元素

|  |
| --- |
|  |
|  |

### 建议使用document.querySelector,因为它里面的参数笔记灵活.实际上它是比较常用的

# 6.可以使用js代码来修改元素的样式,提供元素.style.样式名称=属性值的方式,不过js是使用驼峰命名法,如:css中的background-color在js代码中需要改为元素.style.backgroundColor,而且属性值必须是字符串即使在css中他是整数.这个和css不一样

## 可以参考这个网址: <https://www.w3schools.com/jsreF/dom_obj_style.asp>

# 7.可以使用js来为元素添加类名,因为每一个元素都有一个classList属性,可以在这里添加新的class名称

|  |
| --- |
|  |
|  |
|  |

## 这个classList的作用是,了可以在css中为一个class名称写样式,然后用js代码把这个class名称添加到有需要的元素,这样子实现了,用css写样式然后用js代码动态加载这个样式,也可以将一个class名称从以上的classList中移除

|  |
| --- |
|  |

## 还可以交替使用添加和删除class名称

|  |
| --- |
|  |

# 8.HTML元素有一个innerHTML和一个textContent属性他们是有区别的, innerHTML会返回一个元素类名的使用标签和文本, textContent只能够获取到文本,而且innerHTML也可以设置标签

|  |
| --- |
|  |
|  |

# 9.js获取元素属性,html元素有一个attributes属性,是这个元素的所有属性的集合,我们可以通过js使用元素对象.getAttribute(属性名称)来获取一个属性值,可以使用setAttribute(属性名,属性值)来添加属性.

|  |
| --- |
|  |

# 10.骰子案例:

|  |
| --- |
|  |
| <!DOCTYPE html>  <html lang="en" dir="ltr">    <head>      <meta charset="utf-8">      <title>Dicee</title>      <link rel="stylesheet" href="styles.css">      <link href="https://fonts.googleapis.com/css?family=Indie+Flower|Lobster" rel="stylesheet">    </head>    <body>      <div class="container">        <h1>Refresh Me</h1>        <div class="dice">          <p>Player 1</p>          <img class="img1" src="images/dice6.png">        </div>        <div class="dice">          <p>Player 2</p>          <img class="img2" src="images/dice6.png">        </div>      </div>     <script src="./index.js"></script>    </body>    <footer>      www 🎲 App Brewery 🎲 com    </footer>  </html> |
| /\*styles.css\*/  .container {    width: 70%;    margin: auto;    text-align: center;  }  .dice {    text-align: center;    display: inline-block;  }  body {    background-color: #393E46;  }  h1 {    margin: 30px;    font-family: 'Lobster', cursive;    text-shadow: 5px 0 #232931;    font-size: 8rem;    color: #4ECCA3;  }  p {    font-size: 2rem;    color: #4ECCA3;    font-family: 'Indie Flower', cursive;  }  img {    width: 80%;  }  footer {    margin-top: 5%;    color: #EEEEEE;    text-align: center;    font-family: 'Indie Flower', cursive;  } |
| // alert("inside js!!!");  function rndIndex() {      index = Math.floor(Math.random()\*6)+1;      return index;  }  player1 = document.querySelector(".img1");  index1 = rndIndex();  player1.src="images/dice"+index1+".png";  player2 = document.querySelector(".img2");  index2 = rndIndex();  player2.src="images/dice"+index2+".png";  if(index1 > index2){      document.querySelector("h1").innerHTML="👍Player1 Wins"  } else if(index1 < index2){      document.querySelector("h1").innerHTML="Player2 Wins👍"  }else{      document.querySelector("h1").innerHTML="Draw"  } |

## 玩法:刷新网页,只能够img标签的图片就会实际加载,然后我们根据点数来判断是player1胜利,player2胜利还是平手,我们做了一个小处理,就是序号大的图片的点数多,根据序号就可以判断胜负,效果如下

|  |
| --- |
|  |
|  |
|  |

# 11.js按钮事件监听,我们通过一个Drumkit案例来学习,项目架构如下

|  |
| --- |
|  |

## 1)先把index.js链接到index.html中

|  |
| --- |
|  |

## 2).然后我们在index.js中添加一个测试函数,然后给所有按钮添加事件监听

|  |
| --- |
|  |

## 3)然后了点击任意一个按钮都会触发这个函数

|  |
| --- |
|  |

### 版本1,使用类名来实现播放不同的声音

|  |
| --- |
| let audio = new Audio()  nums = document.querySelectorAll(".drum").length  for(var i=0;i<nums;i++){      document.querySelectorAll(".drum")[i].addEventListener("click",function () {          //console.log(this.classList);          if(this.classList[0]==="w"){              audio.src="sounds/tom-1.mp3";          }else if(this.classList[0]==="a"){              audio.src="sounds/tom-2.mp3";          }          else if(this.classList[0]==="s"){              audio.src="sounds/tom-3.mp3";          }          else if(this.classList[0]==="d"){              audio.src="sounds/tom-4.mp3";          }          else if(this.classList[0]==="j"){              audio.src="sounds/snare.mp3";          }          else if(this.classList[0]==="k"){              audio.src="sounds/crash.mp3";          }          else if(this.classList[0]==="l"){              audio.src="sounds/kick-bass.mp3";          }          audio.play();      })  } |

### 版本2,根据按钮的文本来确定播放哪个声音文件

|  |
| --- |
| let audio = new Audio()  nums = document.querySelectorAll(".drum").length  for(var i=0;i<nums;i++){      document.querySelectorAll(".drum")[i].addEventListener("click",function () {          //console.log(this.classList);          if(this.innerHTML==="w"){              audio.src="sounds/tom-1.mp3";          }else if(this.innerHTML==="a"){              audio.src="sounds/tom-2.mp3";          }          else if(this.innerHTML==="s"){              audio.src="sounds/tom-3.mp3";          }          else if(this.innerHTML==="d"){              audio.src="sounds/tom-4.mp3";          }          else if(this.innerHTML==="j"){              audio.src="sounds/snare.mp3";          }          else if(this.innerHTML==="k"){              audio.src="sounds/crash.mp3";          }          else if(this.innerHTML==="l"){              audio.src="sounds/kick-bass.mp3";          }          audio.play();      })  } |

# 12.js中的对象

|  |
| --- |
|  |

## 对象的构造函数

|  |
| --- |
|  |

## 构造对象

|  |
| --- |
|  |

# 13.用switch...case语句来重构我们的Drumkit案例

|  |
| --- |
| let audio = new Audio()  nums = document.querySelectorAll(".drum").length  for(var i=0;i<nums;i++){      document.querySelectorAll(".drum")[i].addEventListener("click",function () {          switch(this.innerHTML){              case "w":                  audio.src="sounds/tom-1.mp3";                  break;              case "a":                  audio.src="sounds/tom-2.mp3";                  break;              case "s":                  audio.src="sounds/tom-3.mp3";                  break;              case "d":                  audio.src="sounds/tom-4.mp3";                  break;              case "j":                  audio.src="sounds/snare.mp3";                  break;              case "k":                  audio.src="sounds/crash.mp3";                  break;              case "l":                  audio.src="sounds/kick-bass.mp3";                  break;              default: console.log("wrong key");            }          audio.play();      })  } |

# 14.然后我们还可以监听按钮按下事件,这时候监听器需要添加给document,这里代码和上面基本上是一样的我们抽取一个公共函数playSound(key),然后在click事件和按钮事件中调用这个函数

|  |
| --- |
| let audio = new Audio()  nums = document.querySelectorAll(".drum").length  for(var i=0;i<nums;i++){      document.querySelectorAll(".drum")[i].addEventListener("click",function () {          playSound(this.innerHTML)      })  }  //监听键盘按键按下事件  document.addEventListener("keypress",function(e){         playSound(e.key)  })    function playSound(key) {          switch(key){              case "w":                      audio.src="sounds/tom-1.mp3";                      break;                  case "a":                      audio.src="sounds/tom-2.mp3";                      break;                  case "s":                      audio.src="sounds/tom-3.mp3";                      break;                  case "d":                      audio.src="sounds/tom-4.mp3";                      break;                  case "j":                      audio.src="sounds/snare.mp3";                      break;                  case "k":                      audio.src="sounds/crash.mp3";                      break;                  case "l":                      audio.src="sounds/kick-bass.mp3";                      break;                  default: console.log("wrong key");            }          audio.play();  } |

# 15.继续学习js对象,我们可以给对象添加方法

|  |
| --- |
|  |

## 也可以把方法添加到构造函数中

|  |
| --- |
|  |

# 16.js中的回调函数,其实就是作为参数的函数

|  |
| --- |
|  |

## 回调函数的运行原理

|  |
| --- |
|  |

# 17.给页面添加动画,还是以Drumkit案例为例,当用户点击按钮或者按下按键,我们需要添加一点动画效果,在css文件里面有一个.pressed样式.当然只是使用css是没有效果的,因为他不会自动还原,我们需要他在几秒钟后自动还原,就需要使用js的setTimeout函数,在这个函数的处理函数里面把这个样式class移除

|  |
| --- |
|  |

# 扩展,使用js播放声音,注意不能够直接自动播放,需要使用按钮点击事件来实现

## 参考1: <https://www.hangge.com/blog/cache/detail_897.html>

## 参考2: <https://blog.csdn.net/lcc2001/article/details/126995978>

# 扩展2: js函数作为函数的参数

|  |
| --- |
|  |