js正式课第六周

transition 过渡属性

在一定时间内，css属性由起始值向结束值之间实现平滑过渡效果
//width 起始值 100px 结束值 300px
//细分属性 transition-property 过渡属性 all（默认）
//transition-duration 过渡时间
//transition-timing-function:ease(默认) |linear|ease-in-out |ease-in|ease-out 动画运行的速度
//transition-delay 延迟时间
//复合写法 transition:width 1s linear 1s;
//transition:1s;
var oDiv = document.getElementById(“div1”);
oDiv.addEventListener(“webkitTransitionEnd”,function(){//动画结束后触发的事件
console.log(1);
},false);

transform 变换

transform:rotate(-45deg);正->顺时针 负->逆时针
transform:skew(45deg,45deg);
transform:scale(0.5,2);默认值是1，比1小表示缩小，比1大表示放大
transform:translate(100px,-100px); /*第一个参数表示水平方向移动的距离 正往右 负往左 第二个参数表示垂直方向移动的距离 正往下 负的值往上
transition:1s;

transition 动画

transition 适合简单动画效果，需要触发条件（就是让css属性发生改变），只有两种状态（起始状态和结束状态
animation 可以设置复杂的动画效果 ，不需要触发条件 ,可以设置n多种状态
animation动画由两部分组成：1.通过@keyframes设置动画运行的轨迹 (声明动画)2.通过animation来调用帧动画*/
keyframes move{/百分比可以理解成时间的百分比/
0%{}50%{} 100%{}
}

animation的细分属性
animation-name 帧动画的名称
animation-duration 动画运行的时间
animation-timing-function 动画运行的速度 ease(默认)
animation-delay 动画延迟的时间
animation-iteration-count:1(默认)|infinite(无数次) 动画执行的次数
animation-direction:normal（默认） |reverse（反方向） | alternate(交替) | alternate-reverse（反方向交替） animation-play-state:running(运动 默认值)|paused(停止的)
animation-fill-mode:none | forwards(第一帧)| backwards（最后一帧） | both(从第一帧开始停留在最后一帧)

animation:move 1s 2 reverse both
animation:move 1s both;

响应式布局

响应式开发概念

- 根据设备的宽度，自动调整页面的大小，从而在各个设备上呈现出最佳的视觉效果

响应式开发的种类

- 小型的项目或需求页 PC端和移动端共用一套
- 大型的项目例如淘宝 PC端（固定布局）和移动端(移动适配)各用一套

响应式所用的技术

- 视口viewport
- 媒体查询（可以识别设备的特征，从而调整页面的布局）
- 布局时用相对单位 rem 100% em
- flex布局
- 图片处理（max-width/min-width）

视口viewport

概念抽象，设置简单
PC端页面和移动端页面最大差别：设备的宽度不一样
默认移动端页面时按照980宽度渲染，这个宽度还是比移动设备宽度大，为了防止滚动条的

出现，早期各个设备做个缩放处理 缩放的比例 = 设备的宽度/980 ,这样处理后，文字图片都非常模糊看不清

所以需要去更改渲染的宽度（布局视口的宽度），改成设备的宽度渲染，这样页面也不会缩放
<meta name="viewport"
content=”width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0”>

width=device-width 布局视口 = 设备宽度
user-scalable=no 是否允许缩放页面 no禁止 yes 允许
initial-scale=1.0 初始缩放值
maximum-scale=1.0 最大缩放比例
minimum-scale=1.0 最小缩放的比例

媒体查询
根据媒体条件设置不同的样式，从而显示不同的效果
@media all|screen|print and (媒体条件){
div{
css样式
}
}
假如公司里会给你两套图 1000和480宽度的设计稿 这两个宽度就是媒体查询设置的两个主断点 自己还会设置次断点
@media all and(max-width:1000px){

}
@media all and(max-width:480px){

}

移动适配

图片的大小得根据设备的宽度缩放 不能用固定单位 px,而是用相对单位rem
rem是相对于根元素（html）字体大小
得各个设备元素尺寸不一样，只需更改根元素字体大小就可以了，根元素尺寸变化了，元素的尺寸就会发生改变
通过媒体查询更改根元素字体大小会有如下问题：1.设置的是设备宽度的范围，不精准 2.根元素字体大小设置多少合适，不太好设置

- 应该根据各个设备宽度，从而设置各个设备的根元素的字体大小

- 元素的尺寸设置多少rem合适

假设各个设备宽度都是750 ，设计稿宽度也是750,如何设置元素的尺寸
设备宽度 根元素字体大小 元素尺寸
750 100 3.36rem*2.1rem 336px*210px
375 50 3.36rem*2.1rem 168px*105px
414 x

设备宽度/根元素字体大小 = 设计稿宽度(750)/100
414/x = 750/100 x = 55.2
处理dpr为2的设备
设计稿宽度(750)/100 = 设备宽度/根元素字体大小
~(function(desW){
var clientW = document.documentElement.clientWidth;
//desW/100 = clientW/x
//x = clientW/desW*100
document.documentElement.style.fontSize = clientW/desW*100+”px”;
})(750);

大型的项目 PC端一套（固定布局），移动端一套（移动适配）
1.dpr 设备像素比
dpr 1.0 2.0 3.0 普通屏幕 高清屏幕 超高清屏幕 dpr = 物理像素/css像素
iphone3 dpr 1.0 375*667
iphone4 dpr 2.0 320*640 设计稿 640 （早两年）
iphone6 750*1334 物理像素 375*667 css像素 dpr = 2 设计稿是750
iphone6 plus 1242*2208 414*736 dpr=3 设计稿 1242
移动适配 根据设计稿，在各个移动设备上呈现的效果是一样（布局是一样的），但是由于设备的宽度各不一样，图片的尺寸，文字的大小，内容间距等会发生的改变

移动端事件

- click 有300ms的延迟 等看你会不会再按一下或长按
  - fastclick.js解决click延迟问题
    if (‘addEventListener’ in document) {
    document.addEventListener(‘DOMContentLoaded’, function() {
    FastClick.attach(document.body);
    }, false);
    }
    调用之后，还是正常的使用click事件，但是在移动端页面不会再有延迟问题
    document.body.onclick = function(){}

触摸事件 单指事件 touchstart（按下时触发） touchmove（移动时触发） touchend（手指

离开屏幕时触发 ） touchcancel(手机发生意外时触发)

3d

- transform: rotateX(-45deg); 正 往前翻 负 往后翻
- transform:rotateY(-45deg);正 往右 负 往左*/
- transform:rotateZ(100deg);//效果等同于rotate效果，但是这个是在z轴上旋转*/
- z轴永远跟x轴和y轴垂直*/
- transform:translateZ(100px);正往前移动，往前移动离你近了，看上去物体会变大，负的值往后移动，离你远了，看上去物体会变小*/
- scaleZ不能单独使用，得配合其他变形方法一起使用，才能有效果*/
- transform: scaleZ(5) rotateX(45deg);*/
- transform:rotateX(45deg) rotateY(15deg);
- transform:rotate3d(1,0,0,45deg);//*rx ry rz* 向量坐标 *a*角度/
  transition: 1s;

zeptu

- touch事件
  - tap 碰到元素时触发 相当于touchstart
  - singleTap 单击
  - doubleTap 双击
  - longTap 按住元素超过750ms后表示长按
  - swipe(划过), swipeLeft（往左划过）, swipeRight（往右划）, swipeUp（往上划）, swipeDown（往下划）

flex布局

flex-direction:column; /主轴的方向 *row*/
flex-wrap: wrap;/超过父容器时是否换行 默认是*nowrap*不换行 *wrap*表示换行/
/* flex-flow:row wrap;//是flex-direction和flex-wrap的复合写法*/
/*justify-content:space-evenly;*/ /默认往主轴的起始位置对齐/
/* justify-content: center;*/
align-items: center;/设置完高度后不起作用!*/
/*align-content: space-evenly;*处理交叉轴换行的空间的空间，对单行不起作用/

flex核心

flex-grow 扩展比例 当有内容剩余时 ，每个子容器按照这个比例扩展内容
flex-shrink 收缩比例 当有内容溢出时，每个子容器按照这个比例搜索内容

flex-basic 基准值 扩展或收缩内容时，这个值为基准
flex-basic 若值为auto，则以width为基准，其他情况都是以flex-basic值为基准
flex: `<flex-grow> <flex-shrink> <flex-basic>`
默认值 flex:0 1 auto;
flex:auto
flex:2 100px
子容器基准值宽度总共300，父容器设置width是600
剩余内容是300 flex-grow
总共6分 第一个div 占剩余内容的3/6 = 3/6*300 = 150 ->0+150=150
第二个div 占剩余内容的2/6 = 2/6*300 = 100 ->100+100=200
第三个div 占剩余内容的1/6 = 1/6*300 = 50 ->200+50=250

子容器基准值宽度总共300，父容器设置width是200
溢出的内容是100
公式：子容器算上收缩比例的基准值/各个容器算上收缩比例的总基准值 = 求的收缩的内容/溢出的内容
第二个div 2*100/0*3+2*100+3*200 = x/100
200/800 = x/100 x = 25 ->最终宽度是100-25 = 75;

第三个div 3*200/ 0*3+2*100+3*200 = x/100
600/800 = x/100 x = 75 ->最终宽度是200-75 = 125

三栏布局

```
*{margin: 0;padding:0}
.container {width: 100%;display: flex;
flex-wrap: wrap;}
.container>*{
width: 100%;
height: 100px;}
.side1{background: green;}
.side2{background: paleturquoise;}

.main{
background: orange;
}

media all and (min-width: 1001px) {
.main{
flex: 2 0%;
}
.side{
flex:1 0%;
}
```

```
}
@media all and (min-width:750px) and (max-width: 1000px){
.main{
order:-1;
}
.side{
flex:1 0%;
}
}
```