CSS：

1.Border-box把border和padding都纳入总的大小之中（没有margin）IE盒子模型

2.position：absolute会把元素变成块级元素，换行

3.按钮的cursor要设置为pointer

4.transform-style：preserve-3D会保留和呈现子元素的3D位置

5.transform-origin是设置旋转的基点

6.perspective是视距，决定用户与Z轴之间的距离，perspective-origin决定视线的左右和上下位置

7.@font-face自定义自己的字体

IE浏览器：EOT Mozilla浏览器：OTF，TTF  
Safari浏览器：OTF，TTF，SVG  
Chrome浏览器：TTF，SVG

1. -webkit-:chrome safari -moz-:firefox -ms-:IE -o-:opera 兼容按这个顺序最后在写标准的CSS属性
2. cubic-bezier(x1,y1,x2,y2)用于animation自定义自己的速度曲线（cubic-bezier(.81, -0.62, 0.23, 1.92)实现蓄力回弹的速度曲线）
3. animation-fill-mode:控制元素在动画执行前和执行后的样式(forwards将样式停留在最后的状态）
4. translate（100%）相对元素自身的宽高在X轴和Y轴上面移动
5. .icon::before{content:’\f015’;font-family:FontAwesome;}在CSS中直接使用fontawesome的图标
6. img{width:100% height:100%}可以将img整个放入div中
7. box-shadow:0px 0px 1px rgba(0,0,0,1)可以使得阴影四周分布
8. opacity会使内容也透明，rgba可以解决这个问题
9. Localstorage只存储string
10. position: fixed;left: 50%;top: 50%;transform: translate(-50%, -50%);可以宽高自适应居中
11. $("#zkdiv").on("click",".zk",function(){ console.log("on 点击一次"); });解决后append的元素没有监听事件
12. Font: 12px/1.5中的12px是字体大小，1.5\*12px是行高
13. :nth-child(even)偶数子元素，nth-child(odd)奇数子元素
14. ::selection可以自定被选中文字的样式
15. div::before {content:"";position: absolute;border-style: solid;left: -15px;border-width: 8px 16px 8px 0;border-color: transparent red transparent transparent;}在div前面画出小三角形
16. getBoundingClientRect用于获取某个元素相对于视窗的位置集合。集合中有top, right, bottom, left等属性。
17. left: calc(50% - (width/2))也可以实现居中！

24.获取窗口宽度w.width = window.innerWidth || document.documentElement.clientWidth || document.body.clientWidth;（兼容）

25.if(arry)不能判断数组为空，必须用if(array === undefined || array.length == 0)

React:

this.setState：this.state通常是用来初始化state的，this.setstate是用来修改state值的。如果你初始化了state之后再使用this.state，之前的state会被覆盖掉，如果使用this.setState,只会替换掉相应的state值

Redux:

Store：保存数据的容器,createStore传入reducer和最初状态

State：包含所有数据的对象（getState()可以拿到值）

Action：View发出的通知（type属性和payload属性）

Dispatch（调度）：View发出Action的方法（会触发Reducer自动执行）

Reducer：作为state的处理过程，接受action和当前state作为参数，返回新的state（必须是纯函数）

Subscribe（订阅）：设置监听函数，当state改变时执行这个函数（返回函数可解除监听）

Middleware：对Dispatch进行重定义，在发出Action和执行Reducer这两步之间添加其他功能

applyMiddlewares：将所有中间件组成一个数组，增强Dispatch（logger要放最后）

Redux-thunk：使得dispatch可以接受函数为参数

Redux-promise：使得dispatch可以接受Promise对象为参数

异步：发出多种Action，改造State以反映不同的操作状态，用Action Creator来实现自动发送第二个Action，然后用redux-thunk使得dispatch可以接受函数为参数或用Redux-promise使得dispatch可以接受Promise对象为参数

React-Redux：

组件：将所有组件分为UI组件（负责UI的呈现）和容器组件（负责管理数据和逻辑）

Connect：接受mapStateToProps和mapDispatchToProps为参数，从UI组件生成容器组件

mapStateToProps：以state为参数，返回含有props同名属性的对象，建立从外部的State对象到UI组件的props对象的映射关系

mapDispatchToProps：以state为参数，返回含有props同名属性的对象，建立从外部的State对象到UI组件的props对象的映射关系

<provider>：让容器组件可以拿到state

<Switch>:只找到第一个path被匹配的<Route>就立即停止继续匹配，并且只把第一个配到的<Route>渲染出来，带有exact属性的<Route>即使先被匹配到，path属性也必须完全和location相同才会渲染

Demo：<https://github.com/jackielii/simplest-redux-example/blob/master/index.js>

React-Router：

<Router>：作为router组件的容器，拥有history属性（值hashHistory、browserHistory、createMemoryHistory）决定监听浏览器地址变化，解析URL为一个地址对象（browserHistory需要在package.json添加--history-api-fallback）

<Route>：path指定路由的匹配规则（两个规则，第二个规则不会生效），component指定在该地址下加载的组件

{this.props.children}:组件的所有子组件，若存在则加载

通配符：

1. :paramNamep匹配URL的一个部分，直到遇到/?#（this.props.params.paramName可以取出这个值）
2. ()标识URL的这部分可选
3. \*匹配任意字符，直到下一个字符
4. \*\*匹配任意字符，直到下一个/?#

<IndexRoute>：显式指定根路由的子组件，默认情况下加载的子组件（替代{this.props.children || <Home/>）

<Redirect>：用于路由的跳转，访问到一个路由会自动跳到另一个路由

<Link>：React版的<a>元素，生成一个连接，在用户点击后跳转到另一个路由，activeStyle属性设置:active的颜色值，activeClassName则指定当前路由的Class

<IndexLink>：链接要根路由不要使用<Link>（硬要使用则指定onlyActiveOnIdex={true}属性），要使用<IndexLink activeClassName=”active”>来避免链接一直处于active而带有activeStyle或activeClassName的颜色

表单处理：在handleSubmit()函数中实现跳转，第一种方法是browserHistory.push(path)，第二种方法是使用context对象，设置contextTypes:{router:React.propTypes.object}然后this.context.router.push(path)

路由钩子：每个路由都有Enter和Leave两个钩子，使用这两个钩子第一种方法是在onEnter和onLeave中指定一个函数，另一种方法是用withRouter(<App/>)（withRouter要从’react-router’导入）对组件进行“升级”，此时this.props.router有一个setRouterLeaveHooK(this.props.routes,function(){})方法来控制路由跳转，第一个参数是要控制的路由（this.props.routes很有可能是数组），第二个参数是一个函数（返回false不跳转，true直接跳转，返回字符串则浏览器弹出该字符串的confirm，确认跳转，取消不跳转）

ES6:

1. let声明的变量只在所在的代码块（块级作用域）里有效，形成TDZ，只能从声明开始之后的地方使用
2. Const 声明的变量同样只在块级作用域中有效，但是值无法改变且必须立即初始化。Const存复合类型只是所存储的地址不变
3. Let、const、class声明的变量不在属于全局对象（window）的属性
4. 解构赋值：从数组或对象中获取数据的方式更方便简单。可以设置默认值（严格为undefined才取默认值）；解构对象必须同名（使用模式）；已声明的变量进行解构必须用大括号包裹；交换数值、指定函数参数默认值、提取JSON数据很便捷
5. 字符Unicode表示法：\uxxxx；四字节字符使用codePointAt()、fromcodePoint()和at()方式，for(...of...)方法可以遍历
6. 字符串搜索：indexOf()、includes()、 startsWith()、endsWith();字符串重复：repeat();字符串补全:padStart()、padEnd()
7. 模板字符串：`<h1>hello ${str}!</h1>`，空格缩进会被保留，大括号中可以存变量、表达式、执行函数，<%...%>放置JS代码、<%=...%>输出JS表达式
8. 标签模板：过滤HTML字符串、多语言转换、嵌入其他语言，String.raw()对反斜杠转义\转义成\\
9. 正则表达式用u修饰符处理四字节UTF-16编码，大括号可以表示Unicode字符，但是也必须使用u修饰符
10. /^[a-z]$/只匹配a-z字母中的一个(^匹配开头 &匹配结尾)
11. y修饰符：后一次匹配从上一次匹配成功的下一个位置开始，相比于g修饰符隐含一个^头部匹配，有y修饰符就有sticky属性