1. padding: % ； % 是看父宽来决定的；
2. IE怪异盒模型 --标准盒模型

IE 盒模型：总宽度= content + margin (border 和 padding)是算在里面的

标准盒模型：总宽度= contnet+margin+padding+margin

1. border-box 出现的场景

a. Border-box : content-box ; 按标准盒模型解析

b. Border-box : border-box; 按怪异盒模型解析

1. flex 布局

注意：设置flex布局以后，子元素的float、clear和vertical-align 属性将失效

容器的属性:

1. Flex-direction: row| row-reverse | column | column-reverse

2.flex-wrap: nowrap | wrap | wrap-reverse

3.justify-content: flex-start | center | flex-end | space-between | space-around 项目在主轴上如何对齐

4. Flex-flow: 是direction和wrap的缩写 默认为 row nowrap

5. align-items属性 定义项目在交叉轴上如何对齐flex-start | center | flex-end | stretch | baseline

6.align-content 如果有多个线轴的时候的对齐方式；只有一个线轴时，该属性失效

以下是项目属性

1. order: 定义项目的排列顺序；数值越小排列越靠前，默认为0；
2. Flex-grow:定义元素的放大比例；默认为0；即如果存在剩余空间也不放大
3. Flex-shrink:定义项目的缩小比例；默认为1；当空间不足，但是一个项目的该属性为0；该元素不受缩小。
4. Flex-basis属性：在分配多余空间之前，项目占据的主轴空间。

align-self: 允许单个项目与其他项目不一样的属性；（单独修改）

#获取元素到顶部的距离

Js: offsetTop offsetLeft

Jq: $().offset().top $().offset().left

# 深拷贝和浅拷贝

Js 对象分为基本类型和复合（引用）类型，基本类型放在栈内存，复合类型放在堆内存。

堆内存 用于存放由new 创建的对象，栈内存放一些基本类型和对象的引用变量

**如果只是简单的变量，内存小，我们直接复制不会发生引用；**

Var a = 123;var b = a; a = 123456; console.log(a);//123456; console.log(b); //123

**但是如果是对象这种内存占用比较大的来说，直接让复制的东西等于要复制的。那么就会发生引用：就是两个都同时指向了一个空间，改变其中一个，另一个也会改变。这就是发生了引用。引用只会发生在对象身上。**

ES6关于数组的复制方法；不会发生引用

1. array.from(要复制的数组)
2. “...”方法 [...要复制的数组]
3. 通过循环复制

关于对象的深拷贝方法：

1. JSON.parse(JSON.stringify(obj))
2. Object.create()

```

/\* ================ 深拷贝 ================ \*/

function deepClone(initalObj, finalObj) {

    var obj = finalObj || {};

    for (var i in initalObj) {

        var prop = initalObj[i];

        // 避免相互引用对象导致死循环，如initalObj.a = initalObj的情况

        if(prop === obj) {

            continue;

        }

        if (typeof prop === 'object') {

            obj[i] = (prop.constructor === Array) ? [] : Object.create(prop);

        } else {

            obj[i] = prop;

        }

    }

    return obj;

```

1. 递归拷贝

function deepClone(initalObj, finalObj) {

    var obj = finalObj || {};

    for (var i in initalObj) {

        if (typeof initalObj[i] === 'object') {

            obj[i] = (initalObj[i].constructor === Array) ? [] : {};

            arguments.callee(initalObj[i], obj[i]);

        } else {

            obj[i] = initalObj[i];

        }

    }

    return obj;

}