1. Flex弹性盒模型

2009年，W3C 提出了一种新的方案----Flex 布局，可以简便、完整、响应式地实现各种页面布局。

|  |  |
| --- | --- |
| **作用在flex容器上** | **作用在flex子项上** |
| flex-direction | order |
| flex-wrap | flex-grow |
| flex-flow | flex-shrink |
| justify-content | flex-basis |
| align-items | flex |
| align-content | align-self |

1. flex-direction

flex-direction用来控制子项整体布局方向，是从左往右还是从右往左，是从上往下还是从下往上。

|  |  |
| --- | --- |
| **取值** | **含义** |
| **row** | 默认值，显示为行。方向为当前文档水平流方向，默认情况下是从左往右。 |
| **row-reverse** | 显示为行。但方向和row属性值是反的 |
| **column** | 显示为列 |
| **column-reverse** | 显示为列。但方向和column属性值是反的 |

1. flex-wrap

flex-wrap用来控制子项整体单行显示还是换行显示。

|  |  |
| --- | --- |
| **取值** | **含义** |
| **nowrap** | 默认值，表示单行显示，不换行。 |
| **wrap** | 宽度不足换行显示 |
| **wrap-reverse** | 宽度不足换行显示，但是是从下往上开始，也就是原本换行在下面的子项现在跑到上面。 |

flex-flow属性是flex-direction和flex-wrap的缩写，表示flex布局的flow流动特性。第一个值表示方向，第二个值表示换行，中间用空格隔开。

1. justify-content

justify-content属性决定了主轴方向上子项的对齐和分布方式

|  |  |
| --- | --- |
| **取值** | **含义** |
| **flex-start** | 默认值，表现为起始位置对齐。 |
| **flex-end** | 表现为结束位置对齐。 |
| **center** | 表现为居中对齐。 |
| **space-between** | 表现为两端对齐。between是中间的意思，意思是多余的空白间距只在元素中间区域分配。 |
| **space-around** | around是环绕的意思，意思是每个flex子项两侧都环绕互不干扰的等宽的空白间距，最终视觉上边缘两侧的空白只有中间空白宽度一半。 |
| **space-evenly** | evenly是匀称、平等的意思。也就是视觉上，每个flex子项两侧空白间距完全相等。 |

1. align-items

align-items中的items指的就是flex子项们，因此align-items指的就是flex子项们相对于flex容器在侧轴方向上的对齐方式。

|  |  |
| --- | --- |
| **取值** | **含义** |
| **stretch** | 默认值，flex子项拉伸。 |
| **flex-start** | 表现为容器顶部对齐。 |
| **flex-end** | 表现为容器底部对齐。 |
| **center** | 表现为垂直居中对齐。 |

1. align-content

align-content可以看成和justify-content是相似且对立的属性，如果所有flex子项只有一行，则align-content属性是没有任何效果的

|  |  |
| --- | --- |
| **取值** | **含义** |
| **stretch** | 默认值。每一行flex子元素都等比例拉伸。例如，如果共两行flex子元素，则每一行拉伸高度是50%。 |
| **flex-start** | 表现为起始位置对齐。 |
| **flex-end** | 表现为结束位置对齐。 |
| **center** | 表现为居中对齐。 |
| **space-between** | 表现为两端对齐。 |
| **space-around** | 每一行元素上下都享有独立不重叠的空白空间。 |
| **space-evenly** | 每一行元素都完全上下等分。 |

1. 作用在flex子项上的CSS属性

|  |  |
| --- | --- |
| **取值** | **含义** |
| **order** | 可以通过设置order改变某一个flex子项的排序位置。所有flex子项的默认order属性值是0。 |
| **flex-grow** | 属性中的grow是扩展的意思，扩展的就是flex子项所占据的宽度，扩展所侵占的空间就是除去元素外的剩余的空白间隙。默认值为0。 |
| **flex-shrink** | 属性中的shrink是“收缩”的意思，flex-shrink主要处理当flex容器空间不足时候，单个元素的收缩比例。默认值是1。 |
| **flex-basis** | flex-basis定义了在分配剩余空间之前元素的默认大小。 |
| **flex** | flex属性是flex-grow，flex-shrink和flex-basis的缩写。 |
| **align-self** | align-self指控制单独某一个flex子项的垂直对齐方式。 |

1. Grid网格布局

Grid布局是一个二维的布局方法，纵横两个方向总是同时存在。

|  |  |
| --- | --- |
| **作用在grid容器上** | **作用在grid子项上** |
| grid-template-columns | grid-column-start |
| grid-template-rows | grid-column-end |
| grid-template-areas | grid-row-start |
| grid-template | grid-row-end |
| grid-column-gap | grid-column |
| grid-row-gap | grid-row |
| grid-gap | grid-area |
| justify-items | justify-self |
| align-items | align-self |
| place-items | place-self |
| justify-content |  |
| align-content |  |
| place-content |  |

1. grid-template-columns和grid-template-rows

对网格进行横纵划分，形成二维布局。单位可以是像素，百分比，自适应以及fr单位（网格剩余空间比例单位）。

有时候，我们网格的划分是很规律的，如果需要添加多个横纵网格时，可以利用repeat()语法进行简化操作。

grid-template-rows: 100px auto 25%;

grid-template-columns: 100px 100px 200px 100px;

grid-template-rows: 1fr 1fr 1fr;

grid-template-columns: 1fr 1fr 1fr;

grid-template-rows: repeat(3 , 1fr);

grid-template-columns: repeat(3 , 1fr);

1. grid-template-areas和grid-template

area是区域的意思，grid-template-areas就是给我们的网格划分区域的。此时grid子项只要使用grid-area属性指定其隶属于那个区。

grid-template是grid-template-rows，grid-template-columns和grid-template-areas属性的缩写。

grid-template-areas:

"a1 a1 a1"

"a2 a2 a3"

"a2 a2 a3";

grid-template:

"a1 a1 a1" 1fr

"a2 a2 a3" 1fr

"a2 a2 a3" 1fr

/1fr 1fr 1fr;

1. grid-column-gap和grid-row-gap

grid-column-gap和grid-row-gap属性用来定义网格中网格间隙的尺寸。

CSS grid-gap属性是grid-column-gap和grid-row-gap属性的缩写。

grid-column-gap: 10px;

grid-row-gap: 20px;

grid-gap:20px 10px;

1. justify-items和align-items

justify-items指定了网格元素的水平呈现方式，是水平拉伸显示，还是左中右对齐。align-items指定了网格元素的垂直呈现方式，是垂直拉伸显示，还是上中下对齐。

place-items可以让align-items和justify-items属性写在单个声明中。

|  |  |
| --- | --- |
| **取值** | **含义** |
| **stretch** | 默认值，拉伸。表现为水平或垂直填充。 |
| **Start** | 表现为容器左侧或顶部对齐。 |
| **end** | 表现为容器右侧或底部对齐。 |
| **center** | 表现为水平或垂直居中对齐。 |

1. justify-content和align-content

justify-content指定了网格元素的水平分布方式。align-content指定了网格元素的垂直分布方式。place-content可以让align-content和justify-content属性写在一个CSS声明中。

|  |  |
| --- | --- |
| **取值** | **含义** |
| **stretch** | 默认值，拉伸。表现为水平或垂直填充。 |
| **start** | 表现为容器左侧或顶部对齐。 |
| **end** | 表现为容器右侧或底部对齐。 |
| **center** | 表现为水平或垂直居中对齐。 |
| **space-between** | 表现为两端对齐。 |
| **space-around** | 享有独立不重叠的空白空间。 |
| **space-evenly** | 平均分配空白空间 |

1. 作用在grid子项上的CSS属性

|  |  |
| --- | --- |
| **取值** | **含义** |
| **grid-column-start** | 水平方向上占据的起始位置。 |
| **grid-column-end** | 水平方向上占据的结束位置。（span属性） |
| **grid-row-start** | 垂直方向上占据的起始位置。 |
| **grid-row-end** | 垂直方向上占据的结束位置。（span属性） |
| **grid-column** | grid-column-start + grid-column-end的缩写。 |
| **grid-row** | grid-row-start + grid-row-end的缩写。 |
| **grid-area** | 表示当前网格所占用的区域，名字和位置两种表示方法。 |
| **justify-self** | 单个网格元素的水平对齐方式。 |
| **align-self** | 单个网格元素的垂直对齐方式。 |
| **place-self** | align-self和justify-self的缩写。 |

1. Viewport视口

在移动端viewport视口就是浏览器显示页面内容的屏幕区域。在viewport中有两种视口，分别表示为， visual viewport（可视视口）和layout viewport（布局视口）。

visual viewport 固定大小跟屏幕大小相同，在上面，而layout viewport 可改变大小，在下面。Layout viewport默认大小为980像素，可通过document.documentElement.clientWidth获取。

现代网页需要将layout viewport设置成跟visual viewport等同大小，方便进行网页制作。

1. Viewport设置

通过<meta>标签进行设置，name属性指定viewport值，content属性进行视口配置。

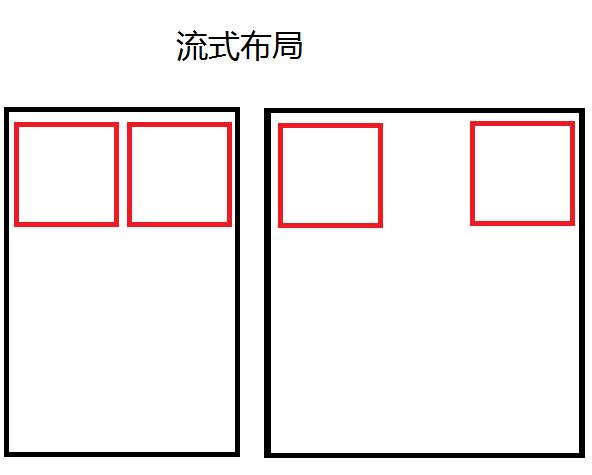
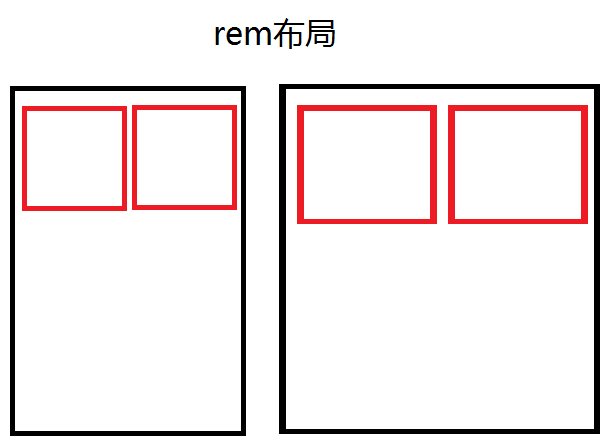
|  |  |
| --- | --- |
| **取值** | **含义** |
| **width** | 设置layout viewport的宽度特定值，device-width表示设备宽。 |
| **height** | 设置layout viewport的高度特定值，一般不进行设置。 |
| **initial-scale** | 设置页面的初始缩放。 |
| **minimum-scale** | 设置页面的最小缩放。 |
| **maximum-scale** | 设置页面的最大缩放。 |
| **user-scalable** | 设置页面能否进行缩放。 |

<meta name="viewport" content="width=device-width, initial-scale=1.0 , user-scalable=no">

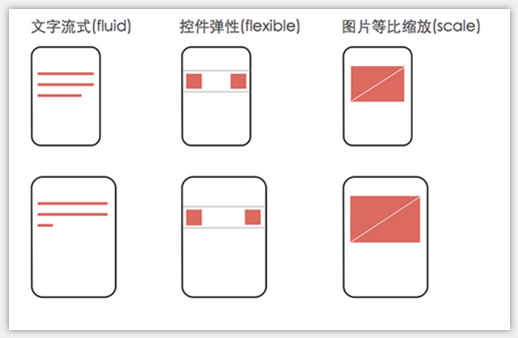
1. 移动端适配方案

1. 百分比布局，也叫流式布局。代表网站：优酷、百度、天猫、腾讯。

2. 等比缩放布局，也叫rem布局。代表网站：网易、爱奇艺、淘宝、美团。

1. 流式布局原则



好处： 大屏幕下显示更多的内容。

坏处：宽屏下比例会有一些不协调。

1. rem布局

单位：

em : 是一个相对单位，1em等于当前元素或父元素的font-size值。

rem : 是一个相对单位，1rem等于根元素的font-size值。

vw / vh : 把屏幕分为100份，1vw等于屏幕宽的1%。

动态设置font-size：

通过JS

通过vw

注：要给body重置一下font-size : 16px

<style>

#box{ width:1.8462rem; height:1rem; background:red;}

</style>

<script>

/\* layout viewport : document.documentElement.clientWidth

iphone 6 : 375

iphone 6 plus : 414

iphone 6 : document.documentElement.clientWidth / 3.75 -> 100 65

iphone 6 plus : document.documentElement.clientWidth / 3.75 -> 110.4 \*/

var fontsize = document.documentElement.clientWidth / 3.75;

document.documentElement.style.fontSize = fontsize + 'px';

</script>

vw : 把屏幕分为了100份，屏幕宽等于100vw ,

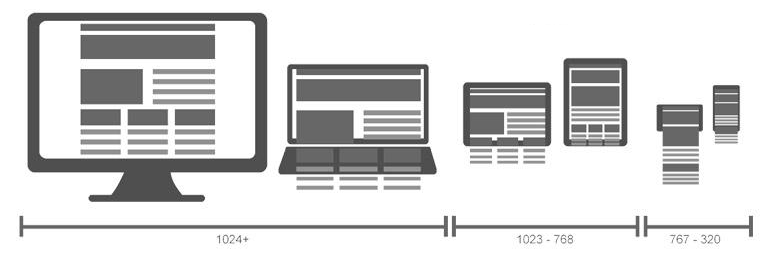
iphone 375 1vw -> 3.75

iphone plus 414 1vw -> 4.14

html{ font-size:26.666667vw;}

1. 响应式布局

利用媒体查询，即media queries，可以针对不同的媒体类型定义不同的样式，从而实现响应式布局 。



媒体类型

|  |  |
| --- | --- |
| **取值** | **含义** |
| all | 用于所有设备。 |
| print | 用于打印机和打印预览。 |
| screen | 用于电脑屏幕，平板电脑，智能手机等。 |
| speech | 应用于屏幕阅读器等发声设备。 |

常见选项：

媒体类型

and、not

min-width、max-width

orientation:portrait、orientation:landscape

<link>

@media all and (min-width:500px) and (max-width:700px){

#box{ background:blue;}

}

@media not all and (min-width:500px){

#box{ background:blue;}

}

@media all and (min-width:500px){

#box{ background:blue;}

}

@media all and (min-width:700px){

#box{ background:yellow;}

}

@media all and ( orientation:portrait ){

#box{ background:blue;}

}

@media all and ( orientation:landscape ){

#box{ background:yellow;}

}

<link rel="stylesheet" href="base.css" media=" all and (min-width:400px) ">

<link rel="stylesheet" href="base2.css" media=" all and (min-width:600px) ">

<link rel="stylesheet" href="base.css" media=" all and (orientation:portrait) ">

<link rel="stylesheet" href="base2.css" media=" all and (orientation:landscape) ">

常见修改样式：display、float、width

注：响应式代码写到要适配的CSS后面。

1. CSS新特性之自定义属性

div{

//自定义颜色

--color : yellow;

}

//root定义的全局常量权限最低

:root{

--color : red;

//自定义px

--number : 100px;

//自定义常量

--number2 : 100;

--size : 50px;

}

#box{

/\* --color : blue; \*/

background: var(--color);

width : var(--number);

//计算

height : calc(var(--number2) \* 1px);

//--size常量有的话使用--size常量，没有使用100px

font-size : var(--size , 100px);

}

1. CSS新特性之shapes

CSS Shapes布局可以实现不规则的文字环绕效果，需要和浮动配合使用。

shape-outside 属性margin-box… polygon

clip-path用于切割 属性polygon(xx xx,xx xx,xx xx)属性用于描点 也可画圆

shape-margin环绕边界距离

1. CSS新特性之scrollbar

CSS scrollbar用于实现自定义滚动条样式。

::-webkit-scrollbar 滑动栏整体

::-webkit-scrollbar-thumb 滑动栏滑动条

::-webkit-scrollbar-track 滑动栏底部条

1. CSS新特性之Scroll Snap

CSS Scroll Snap（CSS 滚动捕捉）允许你在用户完成滚动后多锁定特定的元素或位置。

//父容器设置

scroll-snap-type：x mandatory

//子被滑动容器设置

scroll-snap-align： start / center / end

1. CSS架构

|  |  |
| --- | --- |
| **文件夹** | **含义** |
| **base** | 一些初始的通用CSS，如重置默认样式，动画，工具，打印等。 |
| **components** | 用于构建页面的所有组件，如按钮，表单，表格，弹窗等。 |
| **layout** | 用于布局页面的不同部分，如页眉，页脚，弹性布局，网格布局等。 |
| **pages** | 放置页面之间不同的样式，如首页特殊样式，列表页特殊样式等。 |
| **themes** | 应用不同的主题样式时，如管理员，买家，卖家等。 |
| **abstracts** | 放置一些如：变量，函数，响应式等辅助开发的部分。 |
| **vendors** | 放置一些第三方独立的CSS文件，如bootstrap，iconfont等。 |

.

├── abstracts

│   ├── \_functions.scss

│   ├── \_media.scss

│   └── \_variables.scss

├── base

│   ├── \_animate.scss

│   ├── \_reset.scss

│   ├── \_typography.scss

│   └── \_utilities.scss

├── components

│   ├── \_alert.scss

│   ├── \_button.scss

│   ├── \_form.scss

│   └── \_table.scss

├── layout

│   ├── \_flex.scss

│   ├── \_footer.scss

│   ├── \_grid.scss

│   └── \_header.scss

├── main.scss

├── pages

│   ├── \_detail.scss

│   ├── \_goods.scss

│   ├── \_index.scss

│   └── \_list.scss

├── themes

│   ├── \_admin.scss

│   ├── \_buyer.scss

│   ├── \_seller.scss

│   └── \_tourist.scss

└── vendors

├── boostrap.css

└── iconfont.css