1. px 和 em 的区别

📣 **参考回答：**

`px` 全称 `pixel` 像素，是相对于屏幕分辨率而言的，它是一个绝对单位，但同时具有一定的相对性。因为在同一个设备上每个像素代表的物理长度是固定不变的，这点表现的是绝对性。但是在不同的设备之间每个设备像素所代表的物理长度是可以变化的，这点表现的是相对性

`em` 是一个相对长度单位，具体的大小需要相对于父元素计算，比如父元素的字体大小为80px，那么子元素1em就表示大小和父元素一样为80px，0.5em就表示字体大小是父元素的一半为40px

2. *vw*、*vh* 是什么?

📣 **参考回答：**

vw 和 *vh* 是 *CSS3* 新单位，即 *view width* 可视窗口宽度 和 *view height* 可视窗口高度。1*vw* 就等于可视窗口宽度的百分之一，1*vh* 就等于可视窗口高度的百分之一。

3. 介绍下 *BFC* 及其应用

📣 **参考回答：**

所谓 *BFC*，指的是一个独立的布局环境，*BFC* 内部的元素布局与外部互不影响。

触发 *BFC* 的方式有很多，常见的有：

- 设置浮动
- *overflow* 设置为 *auto*、*scroll*、*hidden*
- *positon* 设置为 *absolute*、*fixed*

常见的 *BFC* 应用有：

- 解决浮动元素令父元素高度坍塌的问题

- 解决非浮动元素被浮动元素覆盖问题
- 解决外边距垂直方向重合的问题

4. 介绍下 *BFC*、*IFC*、*GFC* 和 *FFC*

📣 **参考回答：**

- *BFC*：块级格式上下文，指的是一个独立的布局环境，*BFC* 内部的元素布局与外部互不影响。
- *IFC*：行内格式化上下文，将一块区域以行内元素的形式来格式化。
- *GFC*：网格布局格式化上下文，将一块区域以 *grid* 网格的形式来格式化
- *FFC*：弹性格式化上下文，将一块区域以弹性盒的形式来格式化

5. *flex* 布局如何使用?

📣 **参考回答：**

flex 是 Flexible Box 的缩写，意为"弹性布局"。指定容器display: flex即可。

容器有以下属性：flex-direction，flex-wrap，flex-flow，justify-content，align-items，align-content。

- flex-direction属性决定主轴的方向；
- flex-wrap属性定义，如果一条轴线排不下，如何换行；
- flex-flow属性是flex-direction属性和flex-wrap属性的简写形式，默认值为row nowrap；
- justify-content属性定义了项目在主轴上的对齐方式。
- align-items属性定义项目在交叉轴上如何对齐。
- align-content属性定义了多根轴线的对齐方式。如果项目只有一根轴线，该属性不起作用。

项目（子元素）也有一些属性：order，flex-grow，flex-shrink，flex-basis，flex，align-self。

- order属性定义项目的排列顺序。数值越小，排列越靠前，默认为0。

- flex-grow属性定义项目的放大比例，默认为0，即如果存在剩余空间，也不放大。
- flex-shrink属性定义了项目的缩小比例，默认为1，即如果空间不足，该项目将缩小。
- flex-basis属性定义了在分配多余空间之前，项目占据的主轴空间（main size）。
- flex属性是flex-grow, flex-shrink 和 flex-basis的简写，默认值为0 1 auto。后两个属性可选。
- align-self 属性允许单个项目有与其他项目不一样的对齐方式，可覆盖 align-items 属性。默认值为 *auto*，表示继承父元素的align-items属性，如果没有父元素，则等同于 stretch。

6. 分析比较 *opacity: 0*、*visibility: hidden*、*display: none* 优劣和适用场景。

📣 **参考回答：**

- 结构： display:none: 会让元素完全从渲染树中消失，渲染的时候不占据任何空间, 不能点击， visibility: hidden:不会让元素从渲染树消失，渲染元素继续占据空间，只是内容不可见，不能点击 opacity: 0: 不会让元素从渲染树消失，渲染元素继续占据空间，只是内容不可见，可以点击
- 继承： display: none和opacity: 0：是非继承属性，子孙节点消失由于元素从渲染树消失造成，通过修改子孙节点属性无法显示。 visibility: hidden：是继承属性，子孙节点消失由于继承了hidden，通过设置visibility: visible;可以让子孙节点显式。
- 性能： displaynone : 修改元素会造成文档回流,读屏器不会读取display: none元素内容，性能消耗较大 visibility:hidden: 修改元素只会造成本元素的重绘,性能消耗较少读屏器读取visibility: hidden元素内容 opacity: 0 ： 修改元素会造成重绘，性能消耗较少

7. 如何用 *css* 或 *js* 实现多行文本溢出省略效果，考虑兼容性

📣 **参考回答：**

CSS 实现方式

单行：

```
overflow: hidden;
text-overflow:ellipsis;
white-space: nowrap;
```

多行：

```
display: -webkit-box;
-webkit-box-orient: vertical;
-webkit-line-clamp: 3; //行数
overflow: hidden;
```

兼容：

```
p{position: relative; line-height: 20px; max-height: 40px;overflow:
hidden;}
p::after{content: "..."; position: absolute; bottom: 0; right: 0;
padding-left: 40px;
background: -webkit-linear-gradient(left, transparent, #fff 55%);
background: -o-linear-gradient(right, transparent, #fff 55%);
background: -moz-linear-gradient(right, transparent, #fff 55%);
background: linear-gradient(to right, transparent, #fff 55%);
}
```

JS 实现方式：

- 使用split + 正则表达式将单词与单个文字切割出来存入words
- 加上 '...'
- 判断scrollHeight与clientHeight，超出的话就从words中pop一个出来

8. 居中为什么要使用 *transform*（为什么不使用 *marginLeft/Top*）（阿里）

📣 **参考回答：**

transform 属于合成属性（composite property），对合成属性进行 transition/animation 动画将会创建一个合成层（composite layer），这使得被动画元素在一个独立的层中进行动画。通常情况下，浏览器会将一个层的内容先绘制进一个位图中，然后再作为纹理（texture）上传到 GPU，只要该层的内容不发生改变，就没必要进行重绘（repaint），浏览器会通过重新复合（recomposite）来形成一个新的帧。

top/left属于布局属性，该属性的变化会导致重排（reflow/relayout），所谓重排即指对这些节点以及受这些节点影响的其它节点，进行CSS计算->布局->重绘过程，浏览器需要为整个层进行重绘并重新上传到 GPU，造成了极大的性能开销。

9. 介绍下粘性布局（*sticky*）（网易）

📣 **参考回答：**

position 中的 sticky 值是 CSS3 新增的，设置了 sticky 值后，在屏幕范围（viewport）时该元素的位置并不受到定位影响（设置是top、left等属性无效），当该元素的位置将要移出偏移范围时，定位又会变成fixed，根据设置的left、top等属性成固定位置的效果。

sticky 属性值有以下几个特点：

- 该元素并不脱离文档流，仍然保留元素原本在文档流中的位置。
- 当元素在容器中被滚动超过指定的偏移值时，元素在容器内固定在指定位置。亦即如果你设置了top: 50px，那么在sticky元素到达距离相对定位的元素顶部50px的位置时固定，不再向上移动。
- 元素固定的相对偏移是相对于离它最近的具有滚动框的祖先元素，如果祖先元素都不可以滚动，那么是相对于viewport来计算元素的偏移量

10. 说出 *space-between* 和 *space-around* 的区别？（携程）

📣 参考回答：

这个是 *flex* 布局的内容，其实就是一个边距的区别，按水平布局来说，`space-between`是两端对齐，在左右两侧没有边距，而`space-around`是每个 子项目左右方向的 margin 相等，所以两个item中间的间距会比较大。

11. *CSS3* 中 *transition* 和 *animation* 的属性分别有哪些（哔哩哔哩）

📣 参考回答：

transition 过渡动画：

- *transition-property*：指定过渡的 *CSS* 属性
- *transition-duration*：指定过渡所需的完成时间
- *transition-timing-function*：指定过渡函数
- *transition-delay*：指定过渡的延迟时间

animation 关键帧动画：

- *animation-name*：指定要绑定到选择器的关键帧的名称
- *animation-duration*：动画指定需要多少秒或毫秒完成
- *animation-timing-function*：设置动画将如何完成一个周期
- *animation-delay*：设置动画在启动前的延迟间隔
- *animation-iteration-count*：定义动画的播放次数
- *animation-direction*：指定是否应该轮流反向播放动画
- *animation-fill-mode*：规定当动画不播放时（当动画完成时，或当动画有一个延迟未开始播放时），要应用到元素的样式
- *animation-play-state*：指定动画是否正在运行或已暂停

12. 分析比较 *opacity: 0*、*visibility: hidden*、*display: none* 优劣和适用场景

📣 **参考回答：**

1. display: none (不占空间，不能点击)（场景，显示出原来这里不存在的结构）
2. visibility: hidden（占据空间，不能点击）（场景：显示不会导致页面结构发生变动，不会撑开）
3. opacity: 0（占据空间，可以点击）（场景：可以跟transition搭配）

13. 讲一下*png8*、*png16*、*png32*的区别，并简单讲讲 *png* 的压缩原理

📣 **参考回答：**

PNG图片主要有三个类型，分别为 PNG 8/ PNG 24 / PNG 32。

- `PNG 8`：PNG 8中的8，其实指的是8bits，相当于用2^8（2的8次方）大小来存储一张图片的颜色种类，2^8等于256，也就是说PNG 8能存储256种颜色，一张图片如果颜色种类很少，将它设置成PNG 8得图片类型是非常适合的。
- `PNG 24`：PNG 24中的24，相当于3乘以8 等于 24，就是用三个8bits分别去表示 R（红）、G（绿）、B（蓝）。R(0-255),G(0-255),B(0-255)，可以表达256乘以256乘以256=16777216种颜色的图片，这样PNG 24就能比PNG 8表示色彩更丰富的图片。但是所占用的空间相对就更大了。
- `PNG 32`：PNG 32中的32，相当于PNG 24 加上 8bits的透明颜色通道，就相当于R（红）、G（绿）、B（蓝）、A（透明）。R(0~~255),G(0~~255),B(0~~255),A(0~~255)。比PNG 24多了一个A（透明），也就是说PNG 32能表示跟PNG 24一样多的色彩，并且还支持256种透明的颜色，能表示更加丰富的图片颜色类型。

PNG图片的压缩，分两个阶段：

- `预解析（Prediction）`：这个阶段就是对png图片进行一个预处理，处理后让它更方便后续的压缩。

- `压缩（Compression）`：执行Deflate压缩，该算法结合了 LZ77 算法和 Huffman 算法对图片进行编码。

14. 如何用 *CSS* 实现一个三角形

📣 **参考回答：**

可以利用 border 属性

利用盒模型的 `border` 属性上下左右边框交界处会呈现出平滑的斜线这个特点，通过设置不同的上下左右边框宽度或者颜色即可得到三角形或者梯形。

如果想实现其中的任一个三角形，把其他方向上的 `border-color` 都设置成透明即可。

示例代码如下：

```
<div></div>
div{
width: 0;
height: 0;
border: 10px solid red;
border-top-color: transparent;
border-left-color: transparent;
border-right-color: transparent;
}
```

15. 如何实现一个自适应的正方形

📣 **参考回答：**

方法1：利用 CSS3 的 vw 单位

`vw` 会把视口的宽度平均分为 100 份

```
.square {
 width: 10vw;
 height: 10vw;
 background: red;
}
```

方法2：利用 margin 或者 padding 的百分比计算是参照父元素的 width 属性

```
.square {
 width: 10%;
 padding-bottom: 10%;
 height: 0; // 防止内容撑开多余的高度
 background: red;
}
```

16. 清除浮动的方法

📣 **参考回答：**

- clear 清除浮动（添加空div法）在浮动元素下方添加空div，并给该元素写css样式：`{clear:both;height:0;overflow:hidden;}`
- 给浮动元素父级设置高度
- 父级同时浮动（需要给父级同级元素添加浮动）
- 父级设置成inline-block，其margin: 0 auto居中方式失效
- 给父级添加overflow:hidden 清除浮动方法
- 万能清除法 ::after 伪元素清浮动（现在主流方法，推荐使用）

17. 说说两种盒模型以及区别

📣 **参考回答：**

盒模型也称为框模型，就是从盒子顶部俯视所得的一张平面图，用于描述元素所占用的空间。它有两种盒模型，W3C盒模型和IE盒模型（IE6以下，不包括IE6以及怪异模式下的IE5.5+）

理论上两者的主要区别是二者的盒子宽高是否包括元素的边框和内边距。当用CSS给给某个元素定义高或宽时，IE盒模型中内容的宽或高将会包含内边距和边框，而W3C盒模型并不会。

18. 如何触发重排和重绘?

📣 **参考回答：**

任何改变用来构建渲染树的信息都会导致一次重排或重绘：

- 添加、删除、更新DOM节点
- 通过display: none隐藏一个DOM节点-触发重排和重绘
- 通过visibility: hidden隐藏一个DOM节点-只触发重绘，因为没有几何变化
- 移动或者给页面中的DOM节点添加动画
- 添加一个样式表，调整样式属性
- 用户行为，例如调整窗口大小，改变字号，或者滚动。

19. 重绘与重排的区别?

📣 **参考回答：**

重排: 部分渲染树（或者整个渲染树）需要重新分析并且节点尺寸需要重新计算，表现为重新生成布局，重新排列元素

重绘: 由于节点的几何属性发生改变或者由于样式发生改变，例如改变元素背景色时，屏幕上的部分内容需要更新，表现为某些元素的外观被改变

单单改变元素的外观，肯定不会引起网页重新生成布局，但当浏览器完成重排之后，将会重新绘制受到此次重排影响的部分

重排和重绘代价是高昂的，它们会破坏用户体验，并且让UI展示非常迟缓，而相比之下重排的性能影响更大，在两者无法避免的情况下，一般我们宁可选择代价更小的重绘。

『重绘』不一定会出现『重排』，『重排』必然会出现『重绘』。

20. 如何优化图片

📣 **参考回答：**

1. 对于很多装饰类图片，尽量不用图片，因为这类修饰图片完全可以用 CSS 去代替。
2. 对于移动端来说，屏幕宽度就那么点，完全没有必要去加载原图浪费带宽。一般图片都用 CDN 加载，可以计算出适配屏幕的宽度，然后去请求相应裁剪好的图片。
3. 小图使用 *base64* 格式
4. 将多个图标文件整合到一张图片中（雪碧图）
5. 选择正确的图片格式：

- 对于能够显示 WebP 格式的浏览器尽量使用 WebP 格式。因为 WebP 格式具有更好的图像数据压缩算法，能带来更小的图片体积，而且拥有肉眼识别无差异的图像质量，缺点就是兼容性并不好
- 小图使用 PNG，其实对于大部分图标这类图片，完全可以使用 SVG 代替
- 照片使用 JPEG

21. 你能描述一下渐进增强和优雅降级之间的不同吗?

📣 **参考回答：**

渐进增强 （progressive enhancement）：针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验。

优雅降级 （graceful degradation）：一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。

区别：优雅降级是从复杂的现状开始，并试图减少用户体验的供给，而渐进增强则是从一个非常基础的，能够起作用的版本开始，并不断扩充，以适应未来环境的需要。降级（功能衰减）意味着往回看；而渐进增强则意味着朝前看，同时保证其根基处于安全地带。

22. *CSS3* 新增了那些东西?

📣 **参考回答：**

CSS3 新增东西众多，这里列举出一些关键的新增内容：

- 选择器
- 盒子模型属性：*border-radius*、*box-shadow*、*border-image*
- 背景：*background-size*、*background-origin*、*background-clip*
- 文本效果：*text-shadow*、*word-wrap*
- 颜色：新增 *RGBA*，*HSLA* 模式
- 渐变：线性渐变、径向渐变
- 字体：*@font-face*
- 2D/3D转换：*transform*、*transform-origin*
- 过渡与动画：*transition*、*@keyframes*、*animation*
- 多列布局
- 媒体查询

23. 隐藏页面中的某个元素的方法有哪些?

📣 **参考回答：**

隐藏类型

屏幕并不是唯一的输出机制，比如说屏幕上看不见的元素（隐藏的元素），其中一些依然能够被读屏软件阅读出来（因为读屏软件依赖于可访问性树来阐述）。为了消除它们之间的歧义，我们将其归为三大类：

- 完全隐藏：元素从渲染树中消失，不占据空间。

- 视觉上的隐藏：屏幕中不可见，占据空间。
- 语义上的隐藏：读屏软件不可读，但正常占据空。

完全隐藏

(1) display 属性

```
 display: none;
```

(2) hidden 属性 HTML5 新增属性，相当于 display: none

```
<div hidden></div>
```

视觉上的隐藏

(1) 设置 posoition 为 absolute 或 fixed，通过设置 top、left 等值，将其移出可视区域。

```
position:absolute;
left: -99999px;
```

(2) 设置 position 为 relative，通过设置 top、left 等值，将其移出可视区域。

```
position: relative;
left: -99999px;
height: 0
```

(3) 设置 margin 值，将其移出可视区域范围（可视区域占位）。

```
margin-left: -99999px;
height: 0;
```

语义上隐藏

aria-hidden 属性

读屏软件不可读，占据空间，可见。

```
<div aria-hidden="true"></div>
```