1.对于有多个Scene的WebPlayer发布后，Scene是按着构造时定义的顺序单独加载的，当加载完成Level 0时就会立刻显示，而不会等到所有Scene都加载完才显示，当调用Application.LoadLevel(level)时，必须要在之前判断level是否已经加载完成

if (Application.CanStreamedLevelBeLoaded (levelToLoad)) {

Application.LoadLevel (levelToLoad);

}

对于其它的Player都是运行游戏时，默认显示level0，要加载其它场景，也不用判断是否加载完成

2.当Scene被加载时，它场景中包含的所有资源都是被Preloading，除了Scene0除外，因为它会尽可能快的显示，如果需要做到Scene 0也被Preloading，可以用一个Empty Scene 0去加载它

3. TouchScreenKeyboard.Open()时，可以指定键盘类型、是否输入字符或显示passwork等，不显示字符只对一种键盘有效，且对Android不试用，如果要拿到输入框的大小位置等，不能Open()后就拿这时全是0，要等它完成显示完成再可以获取到

4.对于Transform.scale最大可能的情况下进去均匀的缩放

a.不均匀的缩放会slow render速度。

b.某些组件（Sphere Collider, Capsule Collider, Light and Audio Source等），不均匀缩放并不会变成一个你所期待的椭圆，他仍然是一个圆。

c.当父组件不均匀缩放时，child的Box collider可能会匹配不上它的Mesh，因为不均匀缩放会产生扭曲或裁剪

d.为了性能问题，当不均匀绽放的父类改变rotates并不会去更新子类，当更新完成时，被子类察觉了才会去更新，这就会造成子类突然被改变的感觉！

5. Particle Systems并不会受Transform.scale的影响，要改变它的scale,需要改System’s Particle Emitter, Animator and Renderer

6.Android And IOS阴影限制

a.不支持soft shadows

b.在forward rendering path中只能接收一个directional light能产生阴影

c.在deferred rendering path中能接收所有的directional light产生阴影

7.只有不透明的对象才能投射和接收阴影

a. 使用[Transparent](file:///D:\Program%20Files\Unity\Editor\Data\Documentation\en\Manual\shader-TransparentFamily.html) or Particle shaders将不会再Cast Shadows

b. 使用VertexLit shaders不会receive shadows但是会Cast Shadows

8.Shaders被打包中AssertBundle时，如果AssertBundle被卸载时，Sharder也会被清理掉，这时其它地方用到了会无法显示，所以建议把所以用到的Shader都添加到Edit->Project Settings->Graphics->AlwaysIncludedShaders这个列表就行了

a.在Graphics设置中，可以手动设置哪些Light 或 Fog的sharder是不用使用的以节省包大小，当然设置为自动，则是没有用到的会自动不打包进去

b.可以设置一些Sharder variant来预加载，改变Shader的加载速度

9.当一个Scene View的GameObject的Tag设置为EditorOnly，那么在Build Game的时候，这个对象不会被打包进去，它可以很好的用来做某些测试

10. Hard Shadows比Soft Shadows占用更低的消耗(只会消耗GPU)，但是它的阴影边非常税利，现实情况下你是看不到这种阴影，除非在太空中！当Hard Shadows能满足需要时，用它就可以了

11. perspective aliasing的造成是由于camera perspective造成的，离相机近的阴影会被不规则的放大，然后造成锯齿，可以通过Shadow Cascades来降低perspective aliasing，Cascades越多越耗性能，但是在移动平台Cascades是不被技能的，注意Cascades只对directional light有效

12.a. Vertex lighting性能消耗低，只是在Vertex插入光照的求值，在一些老的显示上被支持

b.Pixel lighting 性能消耗稍高，一般的显卡都支持，它是对屏幕上的每一个Pixel进行运算，所以它支持Normal-mapping, light cookies and realtime shadows等，spotlight shapes and point light highlights look much better when rendered in pixel mode

13. The [Quality Settings](file:///D:\Program%20Files\Unity\Editor\Data\Documentation\en\Manual\class-QualitySettings.html) include a Shadow Distance value，这个也可以用于阴影的优化，对象离相机的距离大于这个值，则不会被渲染阴影(这个值可以用于优化阴影的性能)，但是阴影最好少用，它的性能消耗非常大，可以用烘焙替代而不用实时阴影

14.Vidio Memory = (TotalVideoMemory - ScreenMemory – RenderTextureMemory)

Light Map Memory = VideoMemory \* 1/3

当light map大于这个值时，unity就会自动减少它的resolution直到可以放入light map memory为止，这样会降低阴影的质量

15. Regular frustum culling只是移除Camera看不到的，而Occlusion culling是移除在Camera的View视口内，但是被其它对象给完成遮挡了的对象

16. a. surface shaders 当要与Light交互时使用这个Shader，最终Unity还是自动转换成Vertex and Fragment Shaders 代码

b. Vertex and Fragment Shaders 当不需要与Light交互时使用这个Shader,如ImageEffect，当然与Light交互也可以直接用这个写，但是代码会非常复杂

17.a.ReflectionProbe 在不需要的高细节的地方尽量用low resolution，尽量使用上Culling Mask，把一些不重要的Layer忽略掉

b.Realtime Probe消耗就更高，不到万不得以不要使用，可以从Refresh Mode与Time Slicing两方面优化它的性能，使它尽可能的高效

18.HDR(High Dynamic Range Rendering)

a.优势

高强度区域颜色不会丢失

更好的绽放与发光的支持

减少在低频波段的照明区域

b.劣势

Uses Floating Point buffers (rendering is slower and requires more VRAM)

不支持抗锯齿 (but you can use [Anti-Aliasing image effect](file:///D:\Program%20Files\Unity\Editor\Data\Documentation\en\Manual\script-Antialiasing.html) to smooth out the edges)

不是所有的硬件都支持

19.LOD(Level Of Detail) 当对象离相机远时，显示的细节就少，但是正常情况下它渲染的三角形数依然没变，而LOD可以让你设置不同Level的Mesh来减少在远处的对象所渲染的三角形数，level 0为最大细节，值越大，细节就越少，渲染的消耗也就越小

20.渲染优化

**a**.CPU 尽可能减少可见的object数量，尽可能共享Texture减少drawcall，渲染对CPU的影响主要就是object的数量与需要用到的Texture数量，因为CPU要处理这些并推送给GPU，如果要推送的Texture少，消耗时间自然少，可以把小的Texture合并成大的减少推送次数

**b**.GPU 尽可能的减少模型的面数，它是最影响GPU渲染的性能，注意：Unity的顶点数会大于3D Application中的顶点数，因为如果顶点有UV Normal Color等必须要把顶点分开，这会大大的加大顶点数量。

**c**. Texture Compression and Mipmaps，压缩能减少加载时间和减少内在消耗，能很好的增加渲染的性能，Mipmaps通常情况下都是起用的，当在更小的三角形会用更底分辨率的图片以加块渲染，但是在2D游戏或UI元素中是无效的，它们一直是用1：1的分辨率

**d**. LOD and Per-Layer Cull Distances 可以设置相机的剪切距离使一些远处的对象不再渲染，或者把一些小的对象放到单独的层，然后使用 [Camera.layerCullDistances](file:///D:\Program%20Files\Unity\Editor\Data\Documentation\en\ScriptReference\Camera-layerCullDistances.html)对相机的层的剪切距离单独设置，当然也可以使用LOD技术来减少远处对象的渲染网格数，前提是你要在LOD GROUP组件中添加不同距离的mesh

**e**.Realtime Shadows 会大大的增加CPU(draw call 与处理) 和 GPU(运算)消耗，最好不要使用

**F**.自定义的Sharders 尽量不要每个像素不要超过使用一次sin cos exp pow等耗时的操作，千万不要自定义这些功能函数，使用内置的会快许多，在GPU中half 比float快一倍,fixed比half快一倍，所以要根据情况尽量用最佳的变量类型

**21**.导出fbx时最好使用FBX 2012格式，它可以包含有层次的网格、法线和纹理，也可以重复导入验证你的动画，也可以导出没有网络的动画