

# WebUI插件

WebUI插件支持Windows, Mac, Linux, Android (4.19) 和iOS (4.21) 。同样从版本4.19开始，您必须下载并安装Json Library插件。



史诗游戏从4.24至4.26版本破坏了桌面性能

(请参阅目录)

下载

可以从GitHub的以下地址下载此插件：

<https://github.com/tracerinteractive/UnrealEngine/releases>

4.19.0

Latest release

4.19.0 · 8e4560b

tracerinteractive released this 9 hours ago

Assets 5

HttpLibrary-4.19.zip 20.4 MB

JsonLibrary-4.19.zip 19.3 MB

WebUI-4.19.zip 17.9 MB

每个版本还与所有次要引擎更新兼容，这意味着该插件的4.19.0版本也可以与任何对应的修补程序（如4.19.1或4.19.2）一起使用。

**您必须将GitHub帐户链接到您的Epic Games帐户！**

设置说明：<https://www.unrealengine.com/ue4-on-github>



否则，如果您未使用关联帐户登录，则将收到先前的404错误。

## 安装

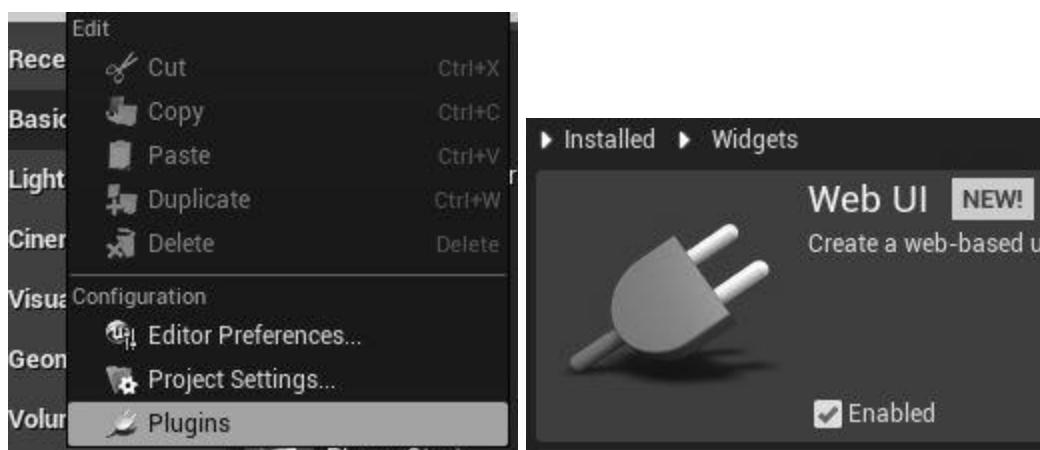
若要安装WebUI插件，请将下载的文件解压缩到以下引擎文件夹中：

This PC > Local Disk (C:) > Program Files > Epic Games > UE_4.19 > Engine > Plugins > Runtime			
Name	Date modified	Type	Size
Synthesis	3/14/2018 5:34 PM	File folder	
WebBrowserWidget	3/14/2018 5:34 PM	File folder	
WebUI	3/14/2018 5:46 PM	File folder	
WindowsDeviceProfileSelector	3/14/2018 5:34 PM	File folder	

另外，请注意屏幕截图中的UE\_4.19目录。您需要将此文件夹更改为与已下载的插件版本相对应的版本。如果您没有将引擎安装到默认目录，请转至您的自定义安装文件夹。

Program Files > Epic Games > UE_4.19 > Engine > Plugins > Runtime > WebUI			
Name	Date modified	Type	
Binaries	3/14/2018 5:46 PM	File folder	
Intermediate	3/14/2018 5:46 PM	File folder	
Source	3/14/2018 5:46 PM	File folder	
WebUI.uplugin	3/14/2018 5:46 PM	UPLUGIN File	

然后打开您的项目并转到编辑下拉菜单中的“插件”选项，单击“小部件”类别，然后启用WebUI插件（如果尚未启用）。



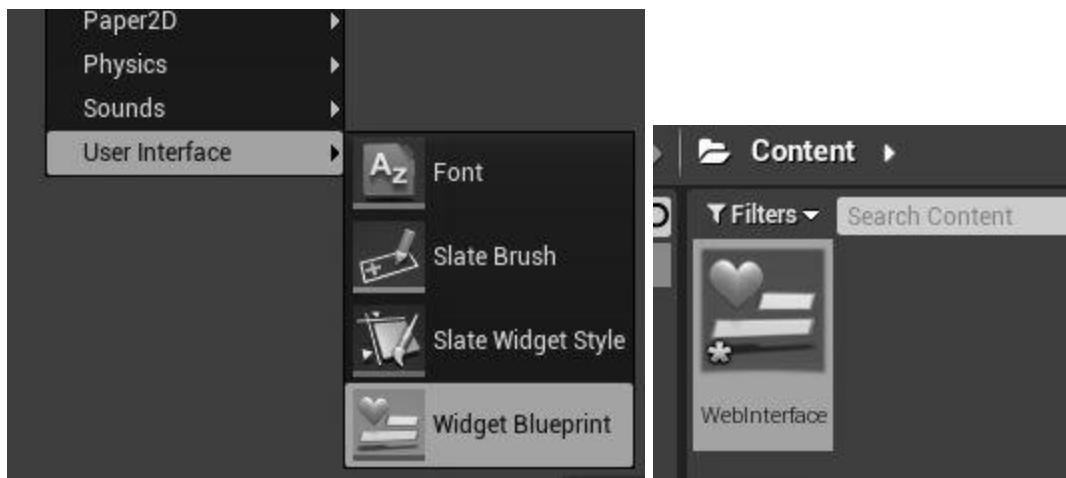
您现在已经成功安装了WebUI插件。重新启动编辑器以继续。

# 目录

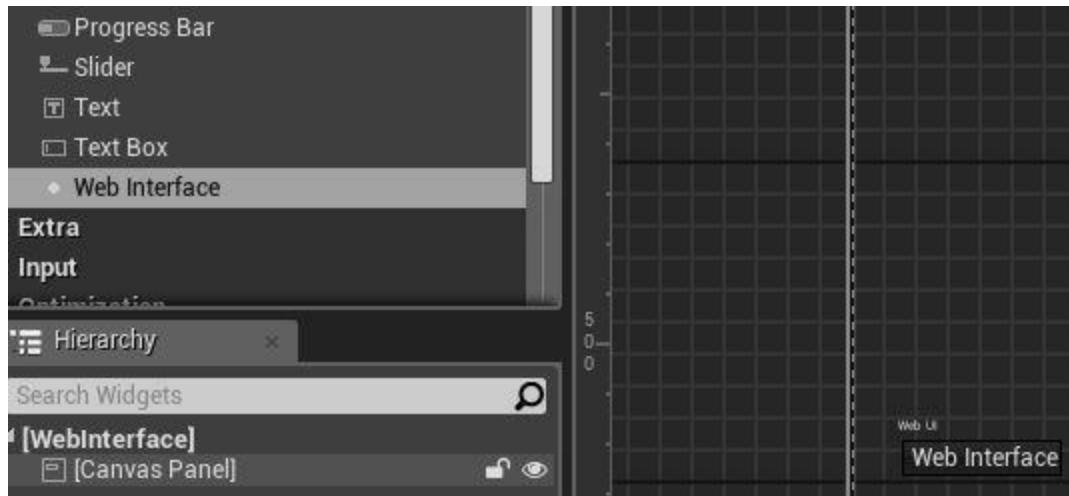
设置	4
数据	13
加载	17
回调	21
聚焦	23
透明度	25
光标	27
纹理	29
工具	30
WEBGL	31
性能表现	32
编译	33

## 设置

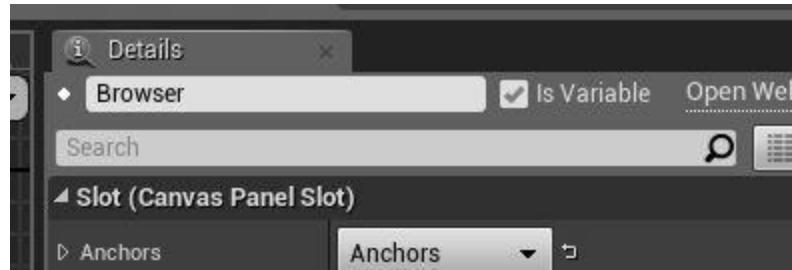
安装并启用WebUI插件后，首先创建一个自定义用户窗口小部件。在此示例中，我们将创建一个称为WebInterface的Web Interface。



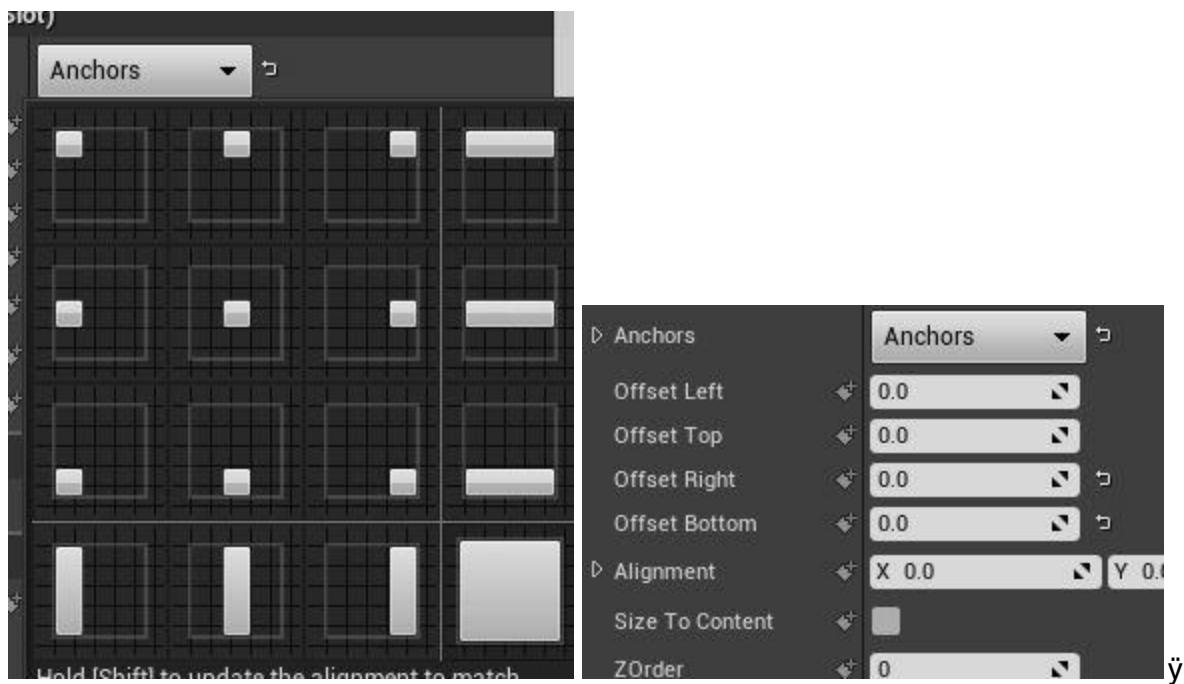
现在打开WebInterface蓝图并开始编辑。将Web Interface组件拖放到画布面板中。



在画布面板中选择Web UI组件，然后设置一个变量名。在此示例中，我们将为该组件使用名称“浏览器”。

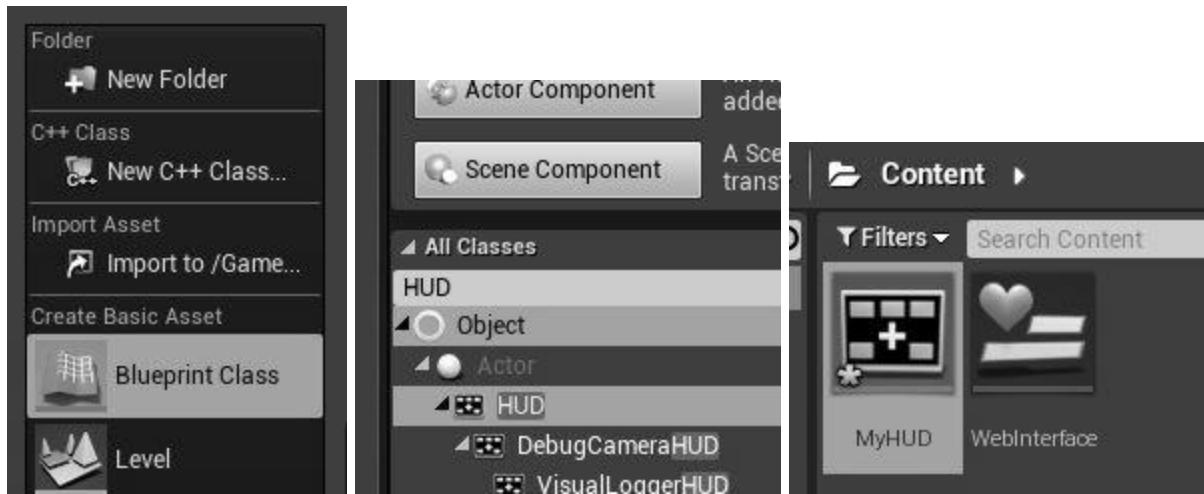


接下来，单击“锚点”下拉列表，然后选择右下角的“snap to all edges”选项，然后将所有偏移设置为零。

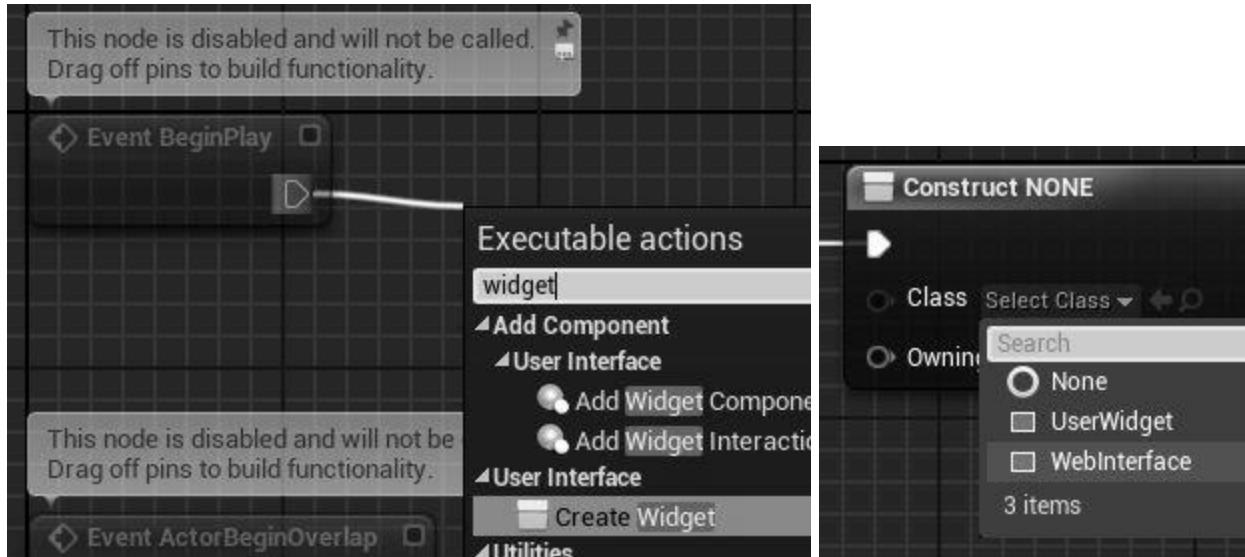


Web UI组件现在应该是全屏的。单击“编译”和“保存”按钮，然后关闭此蓝图。

创建一个新的蓝图类并选择父类。在此示例中，我们将选择HUD类，因为它是最合适的选择，并为此资产使用名称MyHUD。请注意，可以从任何蓝图将小部件添加到视口，因此您可以改用现有的蓝图。



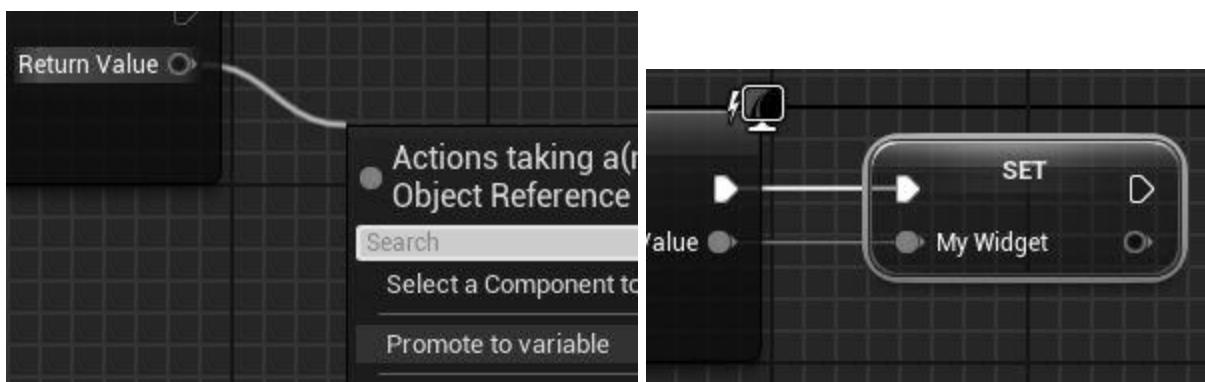
现在打开MyHUD蓝图开始编辑，然后单击“Event Graph”选项卡。从BeginPlay事件中拖动一条执行行，然后选择“Create Widget”节点。然后单击“Select Class”下拉列表，然后选择“Webinterface”窗口小部件。



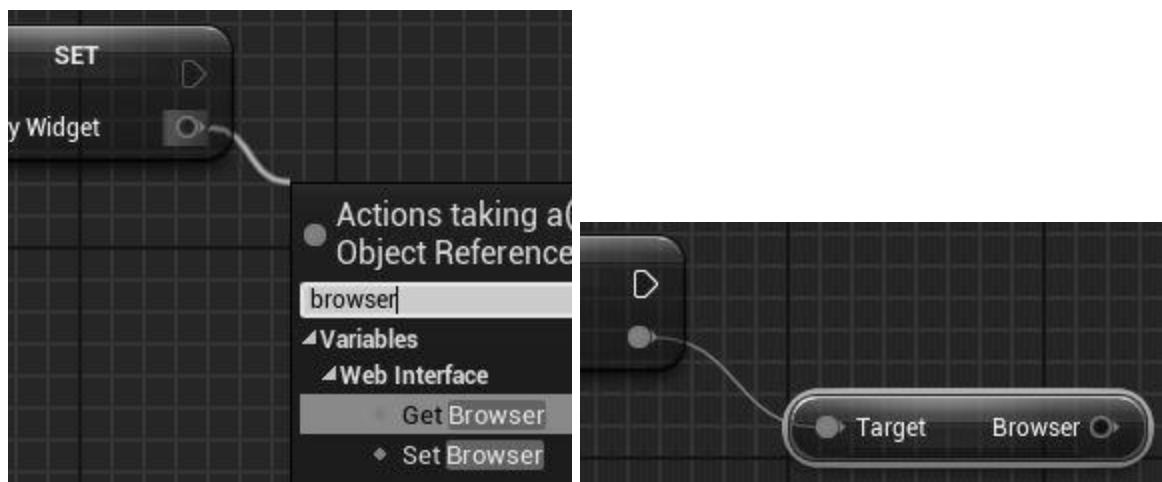
接下来，从“Owning Player”引脚上拖动一个连接，然后选择“Get Owning Player Controller”节点。



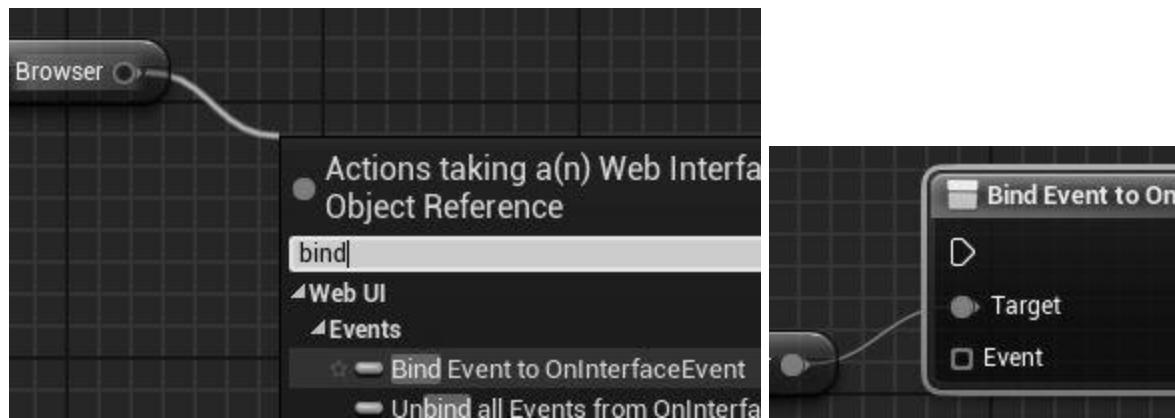
然后从“Return Value”引脚上拖动另一个连接，并在下拉菜单中选择“Promote to variable”选项。



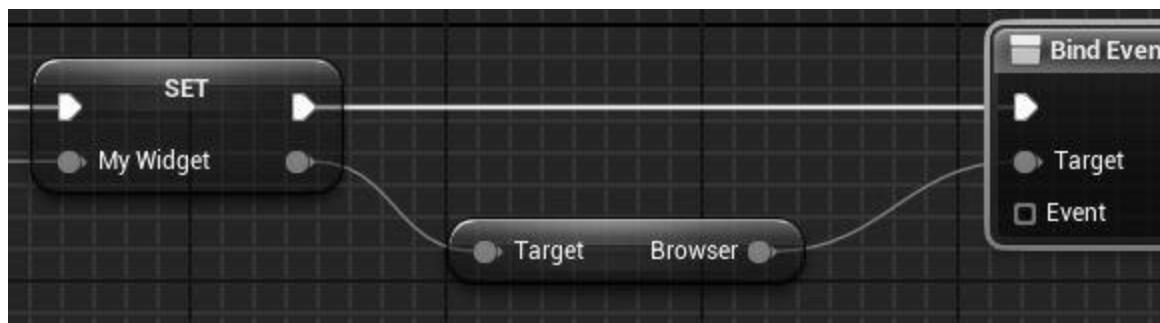
现在，通过从MyWidget引脚拖动连接来读取浏览器变量的值。



接下来，从浏览器变量中拖动一个连接，然后选择“Bind Event to OnInterface Event”节点。



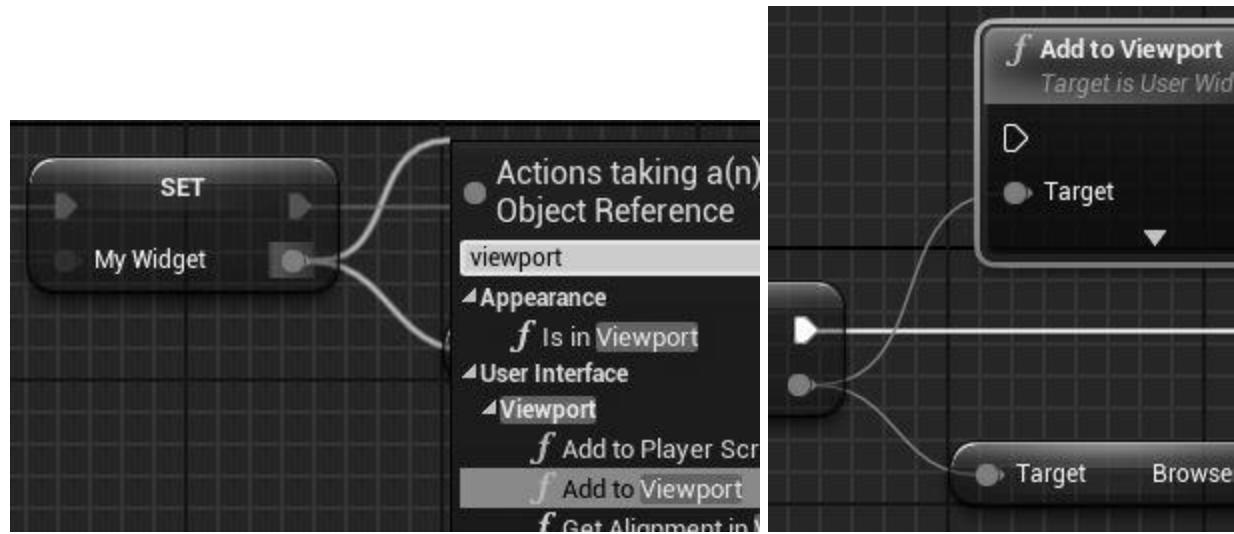
另外，还要确保将执行销钉从MyWidget节点连接到此节点。



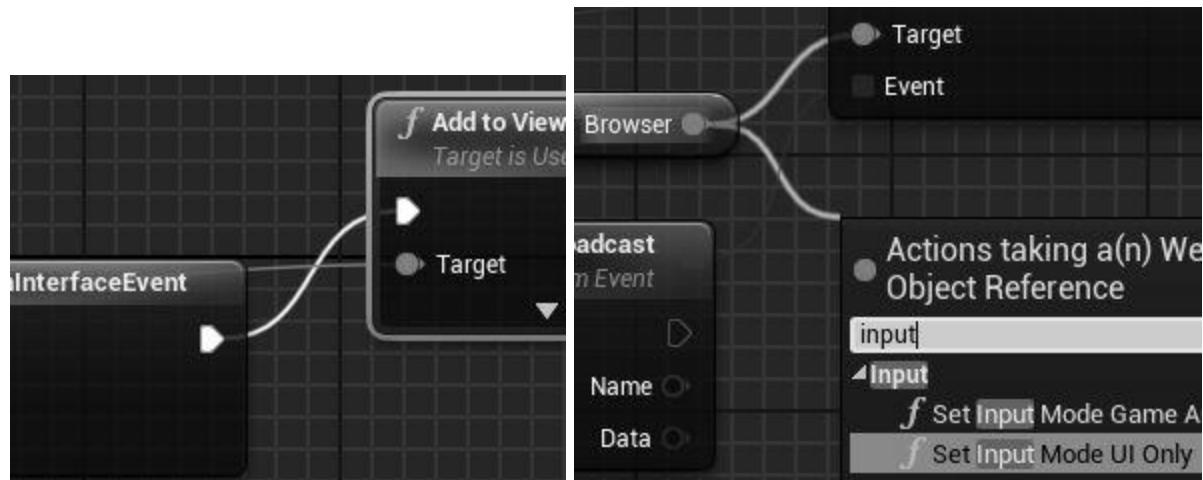
现在，从“Event”图钉中拖动一个委托连接，然后从下拉列表中选择“Add Custom Event..”。在此示例中，我们将为此事件使用名称“OnBroadcast”。



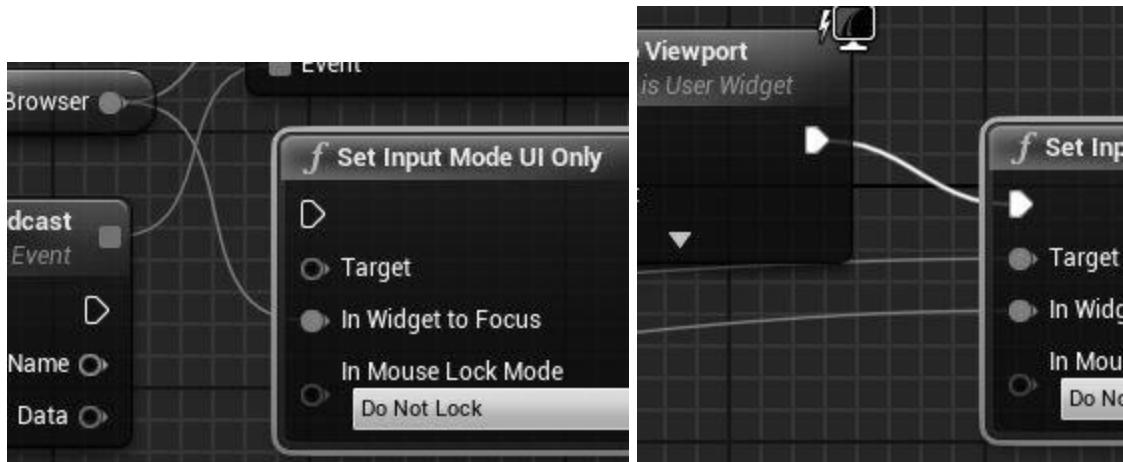
然后从“我的窗口小部件”变量中拖动另一个连接，然后在下拉列表中选择“Add to Viewport”节点。



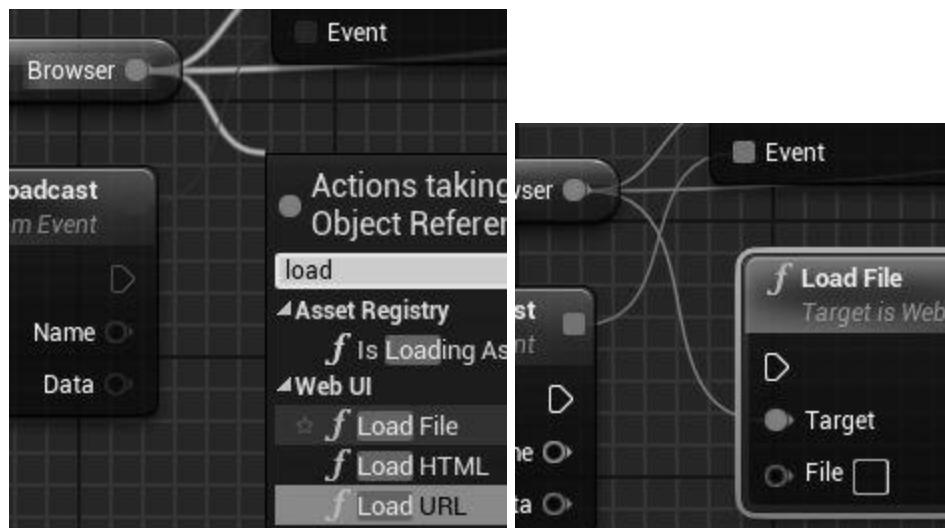
将此节点移到右侧，然后将其连接到“将事件绑定到OnInterfaceEvent”节点。现在，从浏览器变量中拖动一个连接，然后选择“Set Input Mode UI Only”节点。



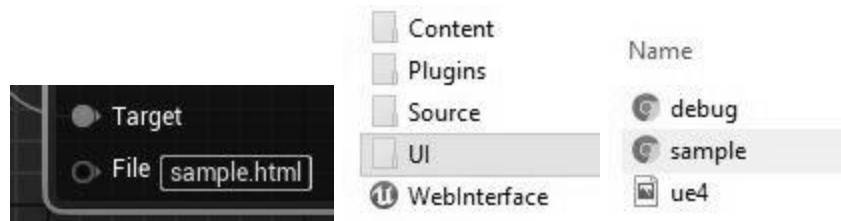
将此节点也向右移动，然后将其连接到“添加到视口”节点。然后，将“Target”引脚连接到先前创建的“获取拥有者播放器控制器”节点。



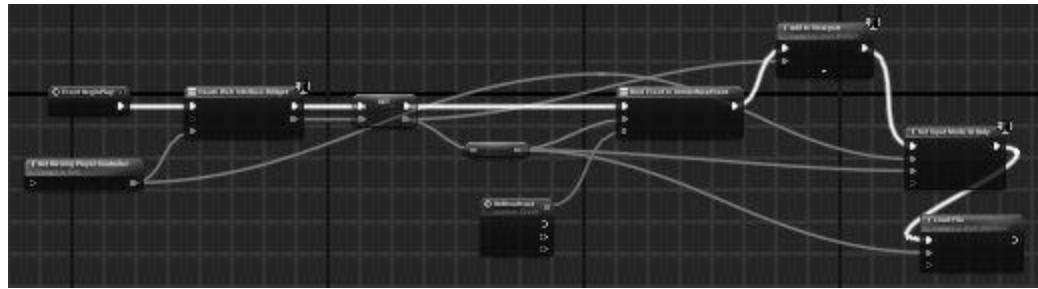
现在，从浏览器变量中拖动另一个连接，然后选择“Load File”节点。



将此节点移到右侧，然后将其连接到“仅设置输入模式UI”节点。然后从/ UI目录输入文件名。尝试使用示例项目中的sample.html文件：

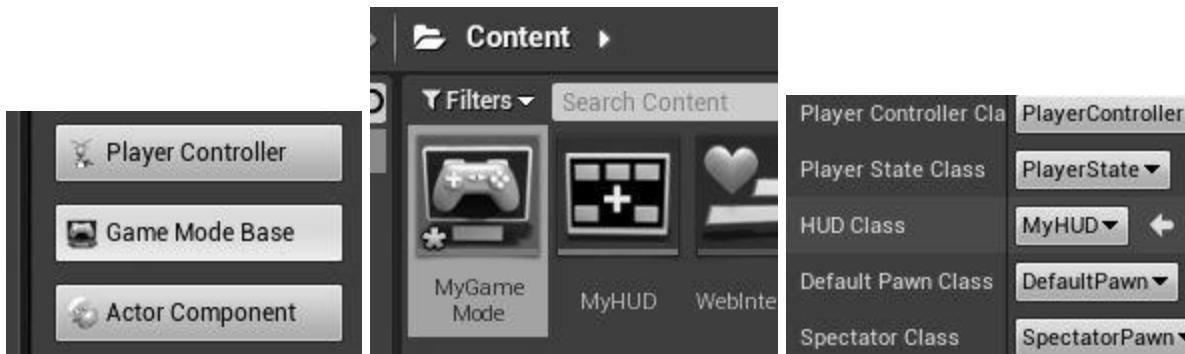


您现在应该拥有一个类似于以下屏幕截图的蓝图：

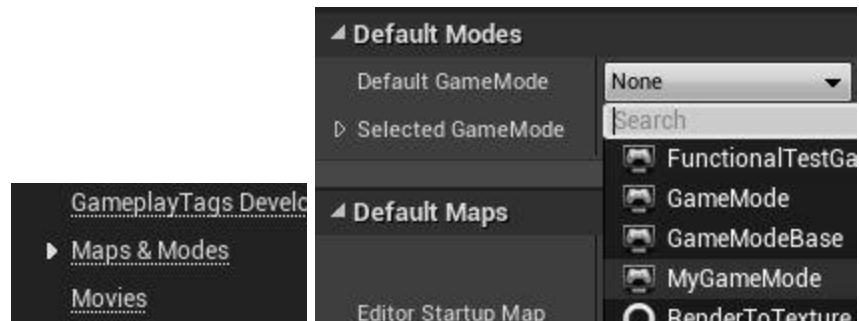


这是WebInterface窗口小部件所需的基线功能。 单击“edit”和“save”按钮，然后关闭此蓝图图。

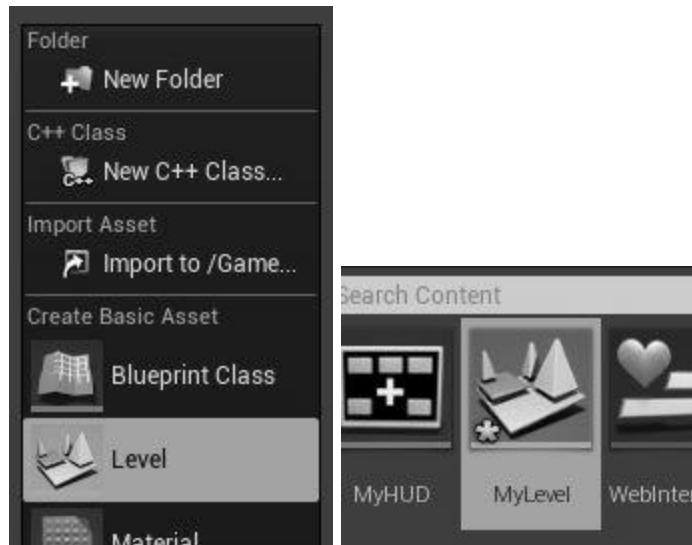
若要在游戏中使用该类，请创建另一个蓝图类，然后选择“Game Mode Base”类作为父类。在此示例中，我们将为该蓝图使用名称My GameMode。 然后在详细信息部分中选择您的HUD。



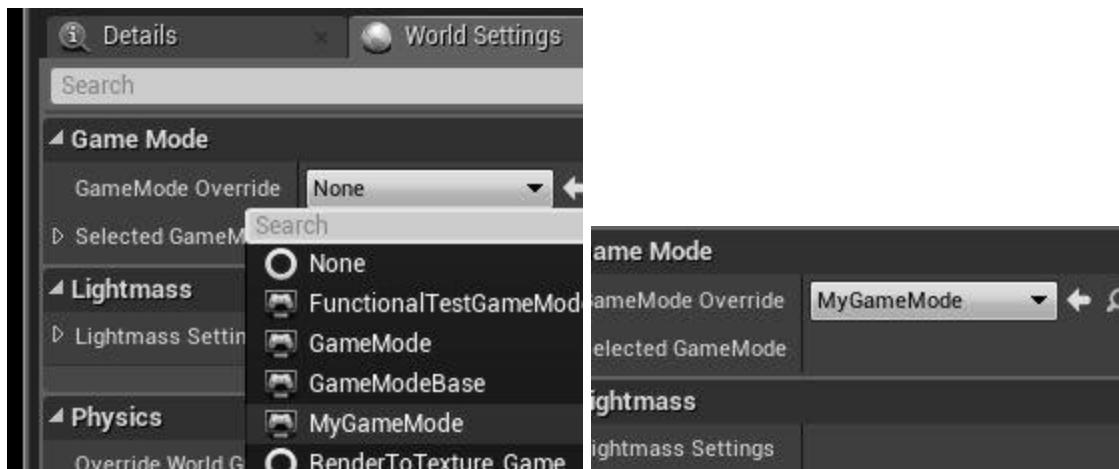
单击“edit”和“save”按钮，然后关闭此蓝图。如果您已有一个要在其中加载界面的级别，请跳过以下部分并改用您自己的自定义级别。您还可以在项目设置中将此游戏模式设置为默认游戏模式，如下所示：



现在创建一个关卡资产来加载您的界面。在此示例中，我们将为此级别使用名称“My Map”。创建完成后，双击级别在编辑器中将其打开。



在“World Settings”选项卡中，从“游戏模式替代”下的下拉列表中选择MyGameMode。然后点击左上角的“Save Current”以保存您的地图。现在您的WebUI已准备好进行测试，只需单击“Play”按钮即可开始！

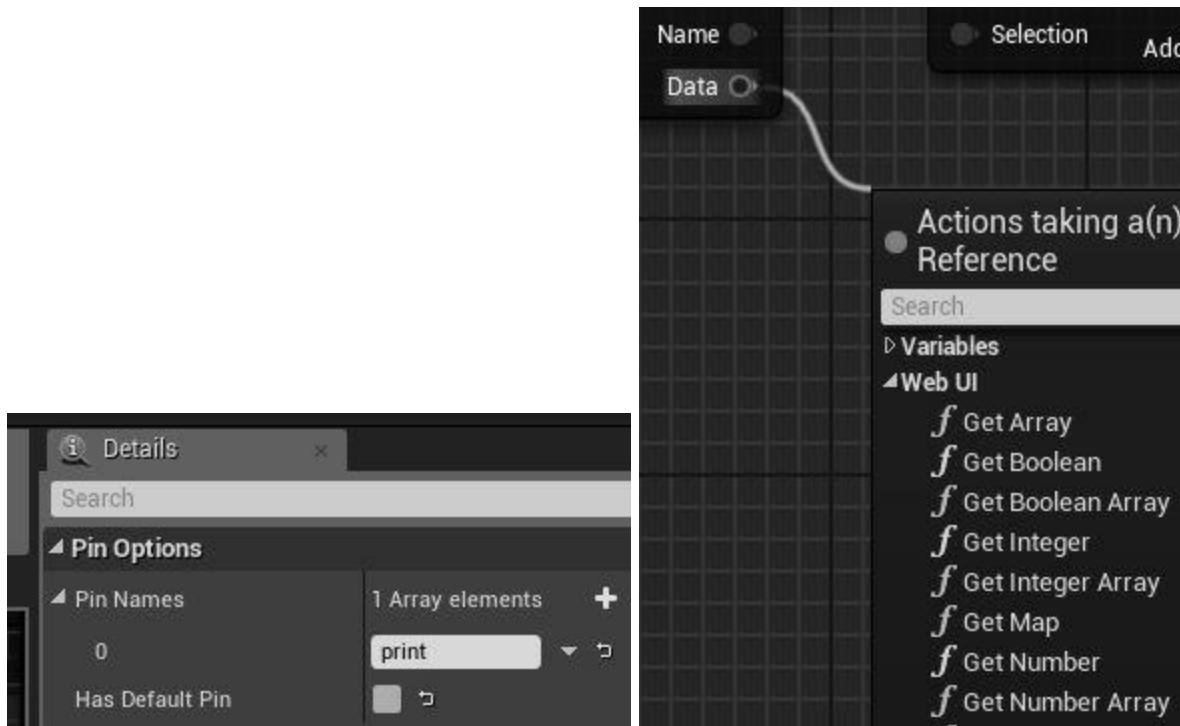


## 数据

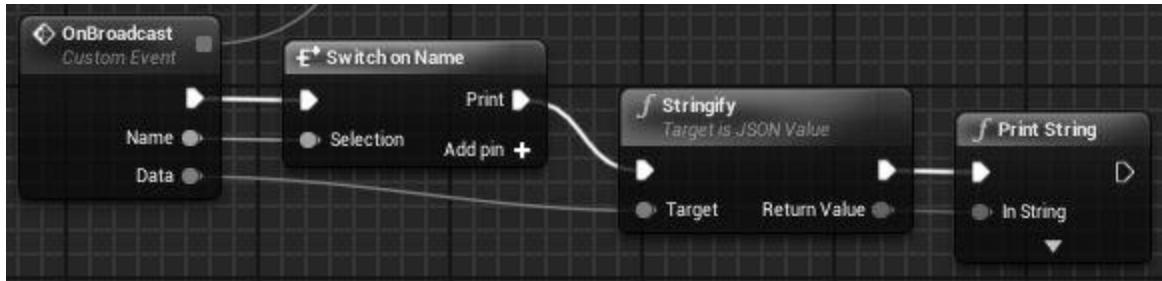
浏览器可以使用自定义蓝图事件将数据作为JSON发送到游戏。首先从先前创建的 OnBroadcast 事件的“Name”针脚拖动连接，然后选择“Switch on Name”节点。



单击“Add pin +”按钮，并选择节点，然后取消选中右侧详细信息面板中的“Has Default Pin”选项。根据所需的功能键入引脚的名称，并根据需要添加更多名称。在此示例中，将使用打印来调试“Data”引脚。从该引脚上拖动连接以遍历和访问从JavaScript发送的对象，数组和原始数据类型。



选择“Stringify”节点并打印“Return Value”引脚后，您的广播事件应类似于以下屏幕截图：



通过调用JavaScript中的`ue.interface.broadcast`函数来触发此事件。第一个参数是事件的“Name”，并且必须是字符串。第二个参数是通过“Data”引脚提供的，并且必须是有效的JSON字符串。全局`ue4 ()`辅助函数已定义为自动`JSON.stringify ...`）第二个参数。

```
jQuery(function()
{
    $("#debugButton").click(function(e)
    {
        var posX = $(this).position().left,
            posY = $(this).position().top;

        // transmit data to the game
        ue4("print", { "posX": posX, "posY": posY });
    });
});
```

此脚本应用于在页面加载时定义全局`ue4 ()`帮助器函数，并且是移动支持所必需的。下页提供了此脚本的源代码。

```
"object"!=typeof ue||"object"!=typeof ue.interface?("object"!=typeof ue&&(ue={}),
ue.interface={},ue.interface.broadcast=function(e,t){if("string"==typeof e){
var o=[e,""];void 0!==t&&(o[1]=t);var n=encodeURIComponent(JSON.stringify(o));
"object"==typeof history&&"function"==typeof history.pushState?(history.pushState(
{},","",#+"+n),history.pushState({},","",#+encodeURIComponent("[]"))):
(document.location.hash=n,document.location.hash=encodeURIComponent("[]"))}):
function(e){ue.interface={},ue.interface.broadcast=function(t,o){
"string"==typeof t&&(void 0!==o?e.broadcast(t,JSON.stringify(o)):
e.broadcast(t,""))}}(ue.interface),ue4=ue.interface.broadcast;
```

`ue.interface.broadcast`功能在Android或iOS上不可用。因此，已开发出一种支持移动浏览器的解决方法。可以在全局帮助器功能的源代码中找到更多详细信息。

`ue4 ()` 帮助程序函数是一个包装器，该包装器使用有效的JSON字符串调用`ue.interface.broadcast`。可选的第二个参数允许在不提供数据的情况下触发命令。此功能还回退到通过移动平台上的URL更改传输JSON。

```
if (typeof ue != "object" || typeof ue.interface != "object")
{
    if (typeof ue != "object")
        ue = {};

    // mobile
    ue.interface = {};
    ue.interface.broadcast = function(name, data)
    {
        if (typeof name != "string")
            return;

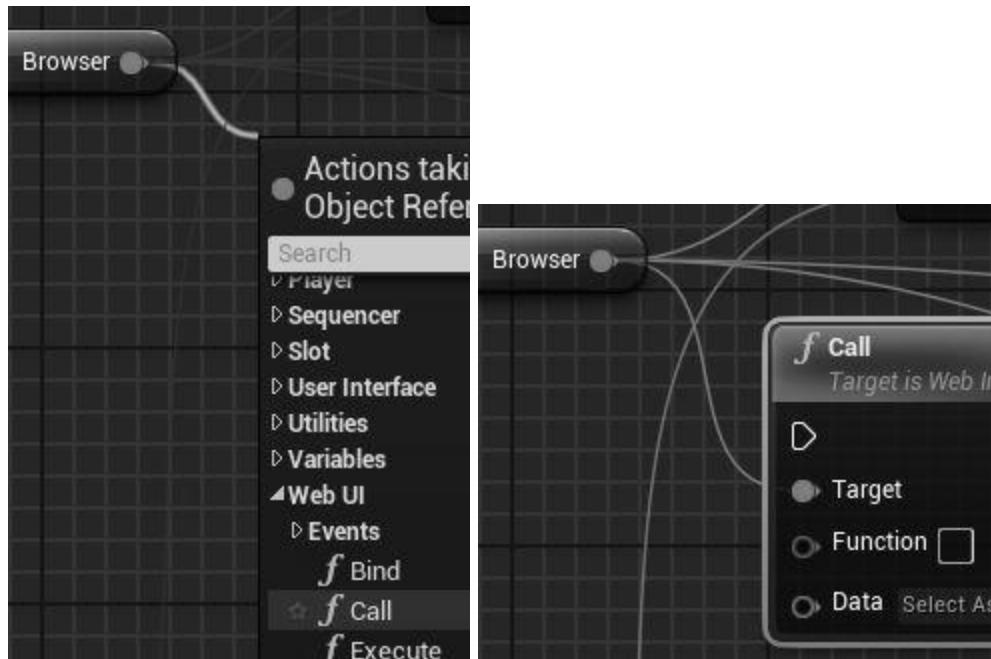
        var args = [name, ""];
        if (typeof data != "undefined")
            args[1] = data;

        var hash = encodeURIComponent(JSON.stringify(args));
        if (typeof history == "object" && typeof history.pushState == "function")
        {
            history.pushState({}, "", "#" + hash);
            history.pushState({}, "", "#" + encodeURIComponent("[]"));
        }
        else
        {
            document.location.hash = hash;
            document.location.hash = encodeURIComponent("[]");
        }
    };
}
else
{
    (function(obj)
    {
        // desktop
        ue.interface = {};
        ue.interface.broadcast = function(name, data)
        {
            if (typeof name != "string")
                return;

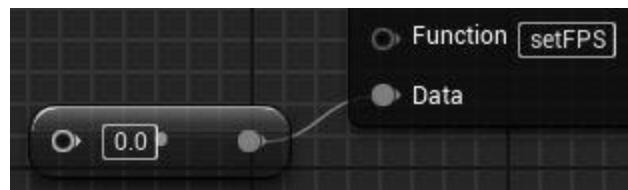
            if (typeof data != "undefined")
                obj.broadcast(name, JSON.stringify(data));
            else
                obj.broadcast(name, "");
        };
    })(ue.interface);

    // create the global ue4(...) helper function
    ue4 = ue.interface.broadcast;
```

游戏还可以通过函数调用以JSON格式将数据发送到浏览器。首先从浏览器变量中拖动一个连接，然后选择“Call”节点。



输入将在JavaScript中执行的函数的名称。在此示例中，将调用`setFPS (...)`函数。现在将JSON值连接到“Data”引脚，可以是浮点数、整数、字符串、布尔值，甚至是复杂类型，例如数组或映射。



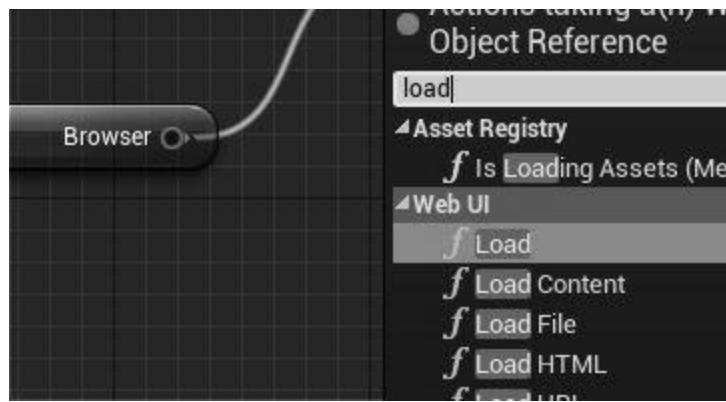
这些函数必须在JavaScript中的全局`ue.interface`对象上定义。提供给“Data”引脚的JSON作为该函数的唯一参数传递，它将自动反序列化为适当的数据类型。

```
// called in-game via blueprints
ue.interface.setFPS = function(fps)
{
    // set element text
    $("#fpsMeter").text(fps.toFixed(1) + " FPS");
};
```

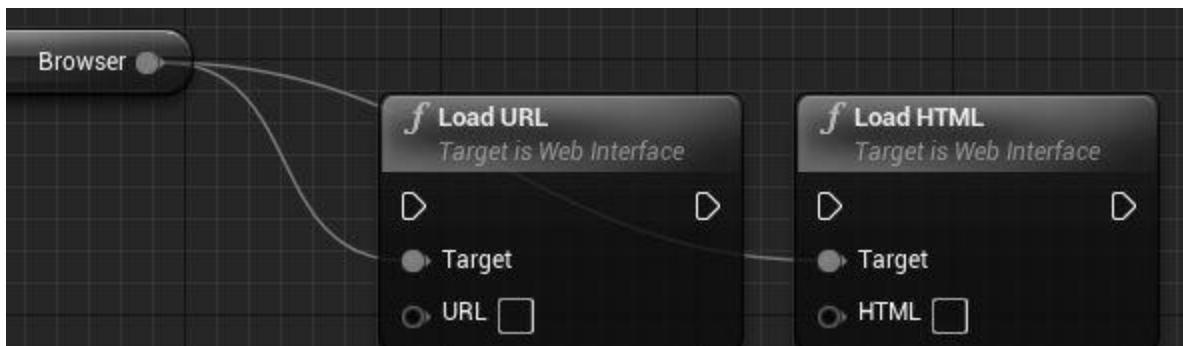
使用这些功能，开发人员可以通过JSON快速轻松地在游戏和浏览器之间发送数据。

## 加载

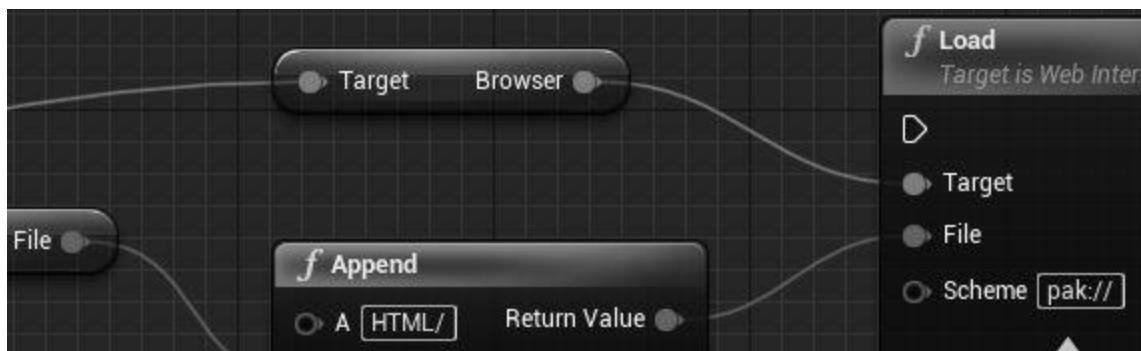
从浏览器变量中拖动连接并选择“Load”功能之一时，浏览器可以加载文件或内容：



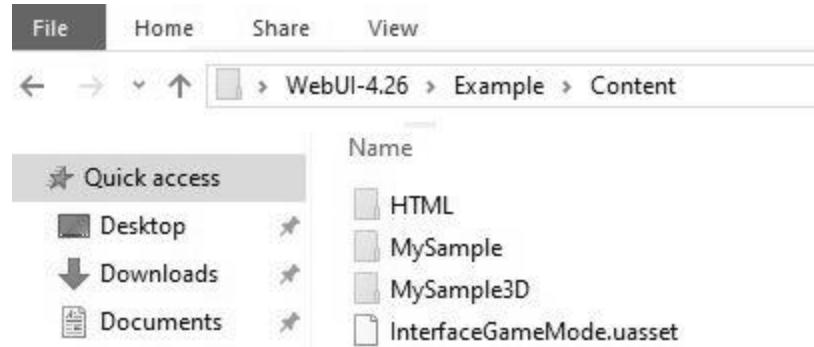
“Load URL”和“Load HTML”节点非常简单。 您可以提供原始HTML字符串，也可以提供将在浏览器中加载的任何URL：



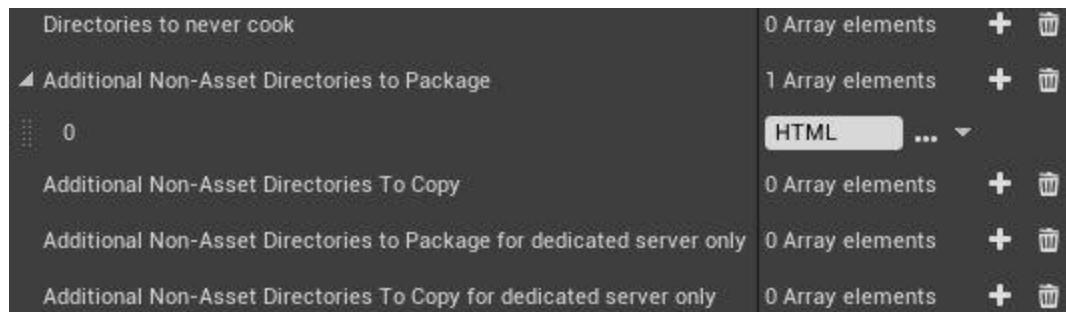
对于大多数游戏，“Load”功能是加载界面的主要方式。 它采用/ Content目录中文件的路径。 该路径将使用自定义的pak://方案重组为URL，该方案使用Unreal Engine文件系统加载内容：



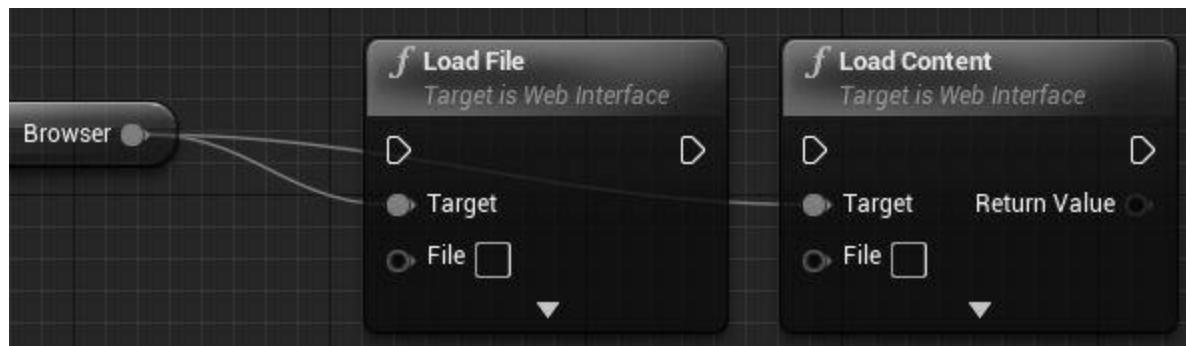
这意味着使用自定义 pak://scheme访问的任何文件都可以直接位于/ Content目录中的磁盘上，以供编辑器构建或打包在内部。运输版本中的pak文件。引擎文件系统将像其他.uasset文件一样自动管理此文件：



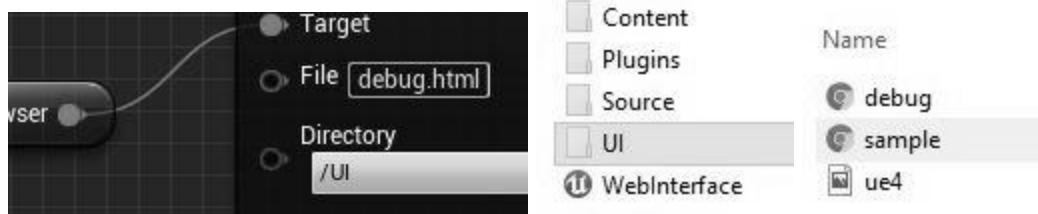
为了确保/ Content目录中的任何文件夹都包含在.pak文件中，请在项目设置中设置“Additional Non-Asset Directories to Package”选项：



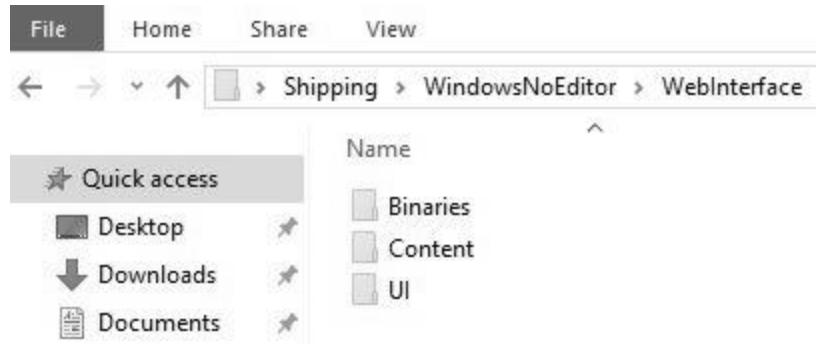
还有“Load File”和“Load Content”节点，它们稍微复杂一些：



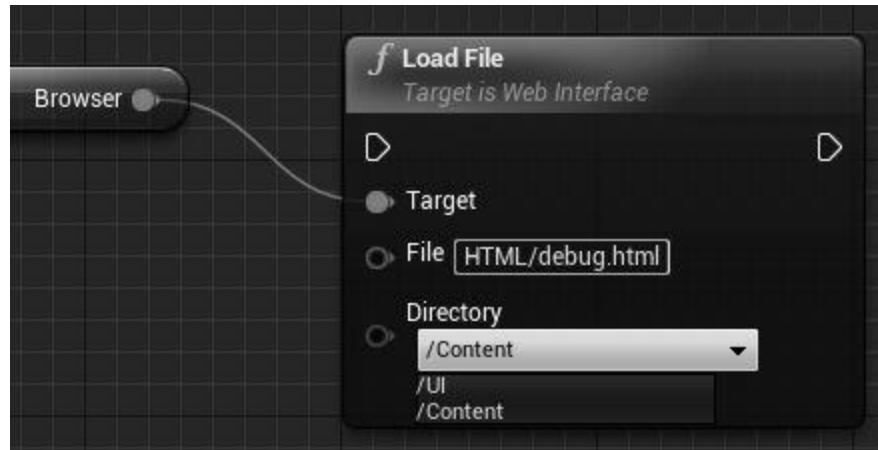
“Load File”功能等效于使用file:/// 并直接在浏览器中加载HTML文件。默认情况下，它将从项目根目录中的IUI目录加载文件：



这也允许使用HTML中的相对路径（例如``）访问图像，脚本和样式表。在运输或包装时，/UI目录应复制到与游戏文件夹中的/Binaries和/Content文件夹相同的级别：

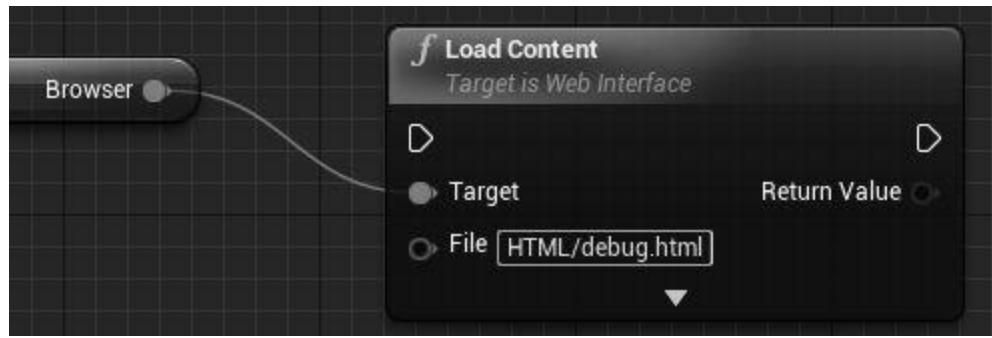


如果您希望将HTML文件放在/Content目录中，那么在高级显示下也可以选择使用此选项。然后，可以使用项目设置中的“Additional Non-Asset Directories To Copy”选项来自动复制/Content目录中的特定文件夹，而不必手动复制UI文件夹：



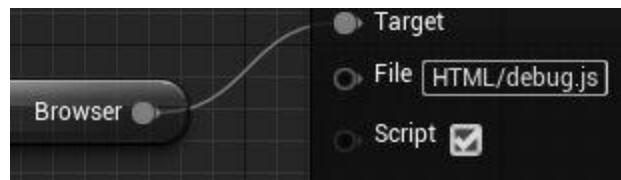
Directories to never cook	0 Array elements
Additional Non-Asset Directories to Package	0 Array elements
Additional Non-Asset Directories To Copy	1 Array elements
0	HTML ...
Additional Non-Asset Directories to Package for dedicated server only	0 Array elements
Additional Non-Asset Directories To Copy for dedicated server only	0 Array elements

但是，“Load File”功能无法访问.pak文件中的HTML文件（即使它们位于 /Content 目录中）。因此，使用“添加到软件包的其他非资产目录”选项添加了“加载内容”功能以访问文件：



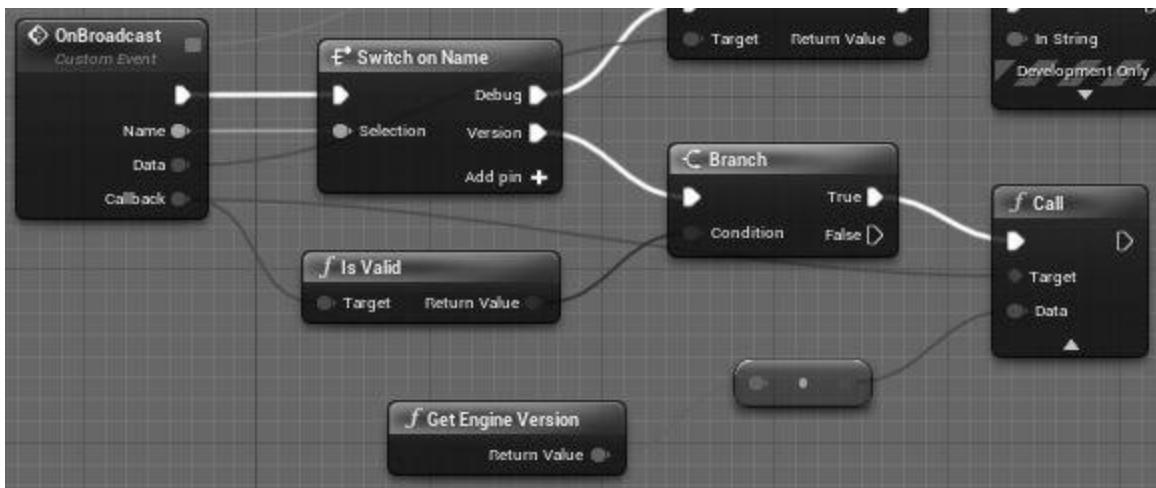
Directories to never cook	0 Array elements
Additional Non-Asset Directories to Package	1 Array elements
0	HTML ...
Additional Non-Asset Directories To Copy	0 Array elements
Additional Non-Asset Directories to Package for dedicated server only	0 Array elements
Additional Non-Asset Directories To Copy for dedicated server only	0 Array elements

此选项将/Content 目录中的文件夹打包到您提供的游戏的pak文件中。但是，由于未使用 file:/// 加载此内容，因此HTML无法访问本地文件，例如图像，脚本和样式表。但是，“Load Content”功能的确在高级显示下提供了一个选项，该选项允许/Content 目录中的 JavaScript文件在浏览器的上下文中执行：



## 回调

从4.24版开始，ue.interface.broadcast函数包含一个可选的第三个参数，该参数必须是字符串。它在ue上指定函数名称。引擎可以选择使用可选参数回调的接口对象。这是一个从蓝图调用回调的示例（如果提供的话）：



可以通过全局ue4()帮助函数从JavaScript触发此事件，并将回调函数作为第二个或第三个参数：

```
ue4("version", function(v)
{
    if (typeof v == "string")
        document.body.innerText = "Unreal Engine " + v.split('-')[0];
});
```

该功能后可以提供一个可选的超时时间。如果未提供超时，则在辅助函数中定义的默认值为1秒。超时后，临时回调函数将被自动删除。

以下是输入数据和3秒钟的回调超时的示例：

```
ue4("version", {}, function(v)
{
    if (typeof v == "string")
        document.body.innerText = "Unreal Engine " + v.split('-')[0];
}, 3);
```

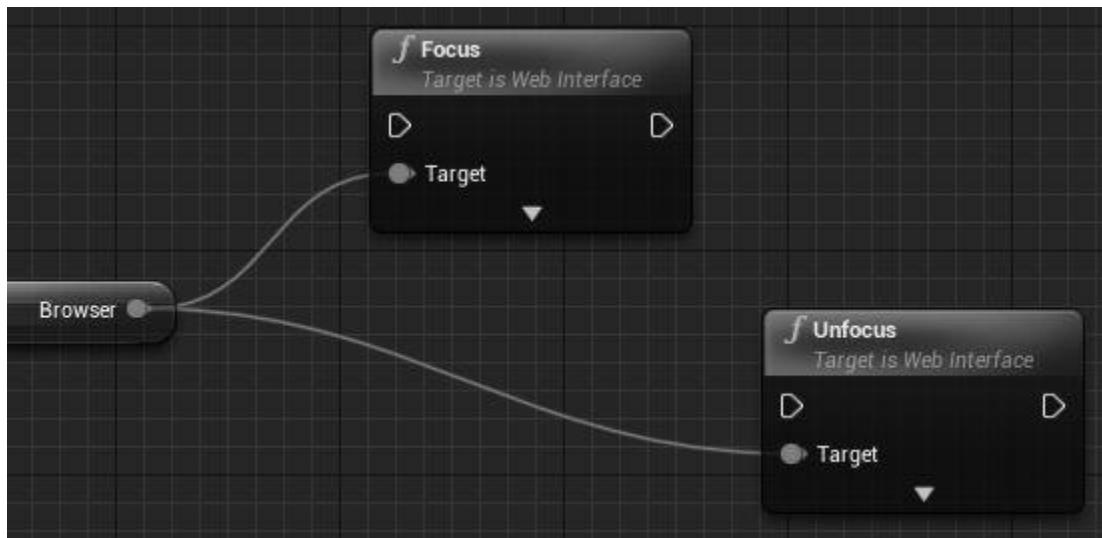
该脚本可用于定义全局ue4()帮助函数，该函数注册具有可选超时期限的临时回调函数。示例项目中提供了此脚本的源代码。

```
"object"!=typeof ue&&(ue={}),uuidv4=function(){  
    return"10000000-1000-4000-8000-100000000000".replace(/[018]/g,function(t){  
        return(t^crypto.getRandomValues(new Uint8Array(1))[0]&15>t/4).toString(16)}),  
    ue4=function(r){return"object"!=typeof ue.interface||"function"!=typeof  
    ue.interface.broadcast?(ue.interface={},function(t,e,n,o){var u,i;"string"==typeof  
    t&&("function"==typeof e&&(o=n,n=e,e=null),u=[t,"",r(n,o)],void 0!==e&&(u[1]=e),  
    i=encodeURIComponent(JSON.stringify(u)),"object"==typeof  
    history&&"function"==typeof history.pushState?(history.pushState({},","",#"+i),  
    history.pushState({},","",#"+encodeURIComponent("[]")):(document.location.hash=i,  
    document.location.hash=encodeURIComponent("[]"))):(i=ue.interface,  
    ue.interface={},function(t,e,n,o){var u;"string"==typeof t&&("function"==typeof  
    e&&(o=n,n=e,e=null),u=r(n,o),void 0!==e?i.broadcast(t,JSON.stringify(e),u):  
    i.broadcast(t,"",u)});var i}(function(t,e){if("function"!=typeof t)return"";  
    var n=uuidv4();return ue.interface[n]=t,setTimeout(function(){delete ue.interface[n]},  
    1e3*Math.max(1,parseInt(e)||0)),n});
```

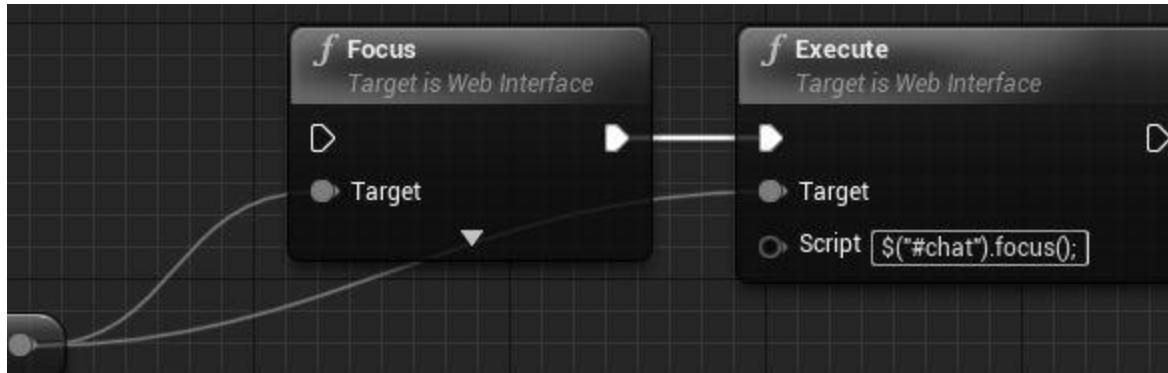
uuidv4函数生成一个加密强度高的UUID，用作ue上的临时函数名称。接口对象。此属性将在设计的超时时间（默认为1秒）后自动删除。

## 聚焦

有“聚焦”和“未聚焦”功能可用于更改键盘和鼠标的聚焦：



“聚焦”节点直接将焦点集中到浏览器的视口中。此方法比引擎提供的默认功能更高级，因为它们要求用户在浏览器获得键盘焦点之前单击界面。但是，插件提供的此功能 允许 JavaScript集中输入和文本框元素。这意味着当与JavaScript焦点事件结合使用时，您 可以将游戏的焦点直接设置为HTML元素（例如，聊天小部件），如下所示：



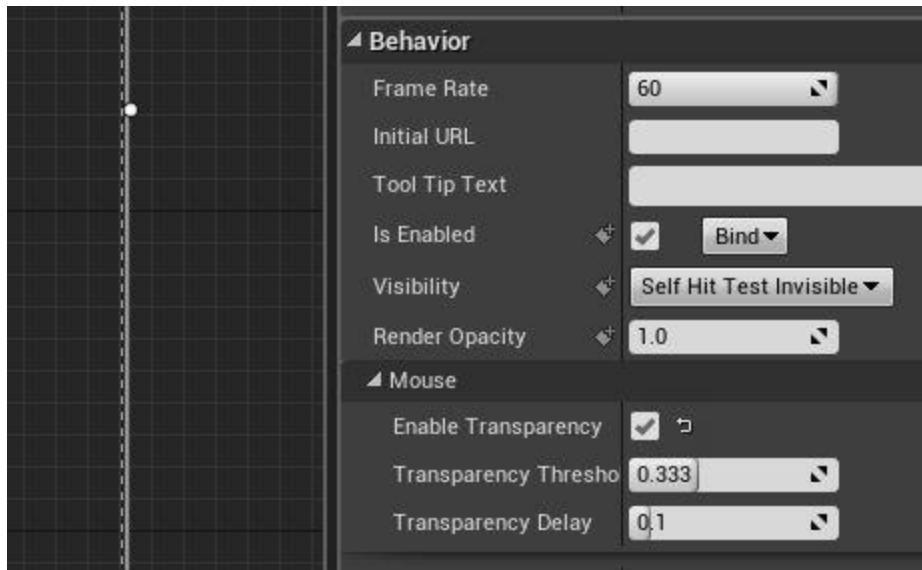
该节点还将自动显示鼠标光标，并使界面完全可交互。然后，“未聚焦”节点将自动隐藏鼠标光标，并将焦点从浏览器移回到游戏上，这两个节点通常用于在游戏和任何游戏内设置之间进行切换（通常在大多数游戏中通过退出键）。

提供了另一个名为“Reset Mouse Position”的节点，以将鼠标（在可见时）移动到屏幕中心。通常在“聚焦”节点之后调用此方法，如果在将焦点设置到界面时最初不可见鼠标，则建议使用此方法：



## 透明度

可以将小部件配置为支持在界面的透明部分后面单击，以进行需要始终显示鼠标光标的游  
戏。此设置在“Behavior”部分下的小部件蓝图中可用：

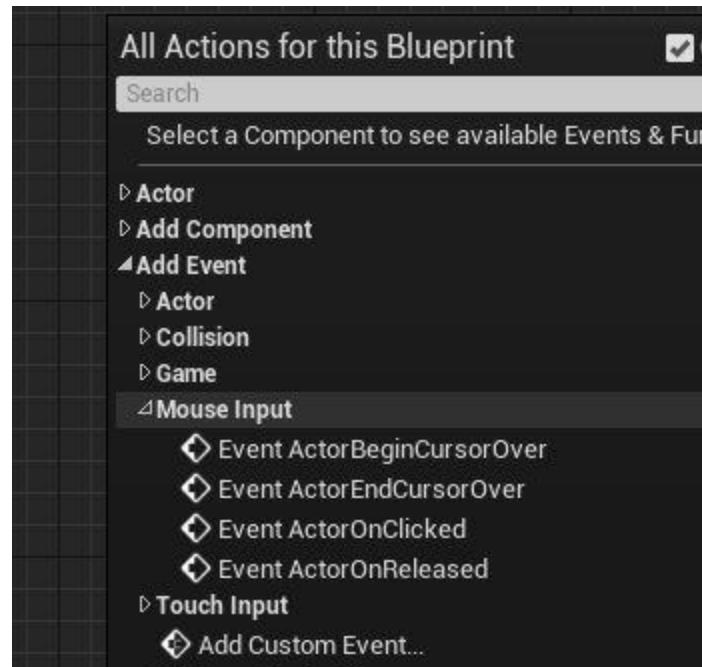


这将指导基础板岩小部件在每个游戏刻度上的鼠标位置下对像素进行采样，并在“点击测  
试不可见”和“可见”模式之间动态切换小部件。

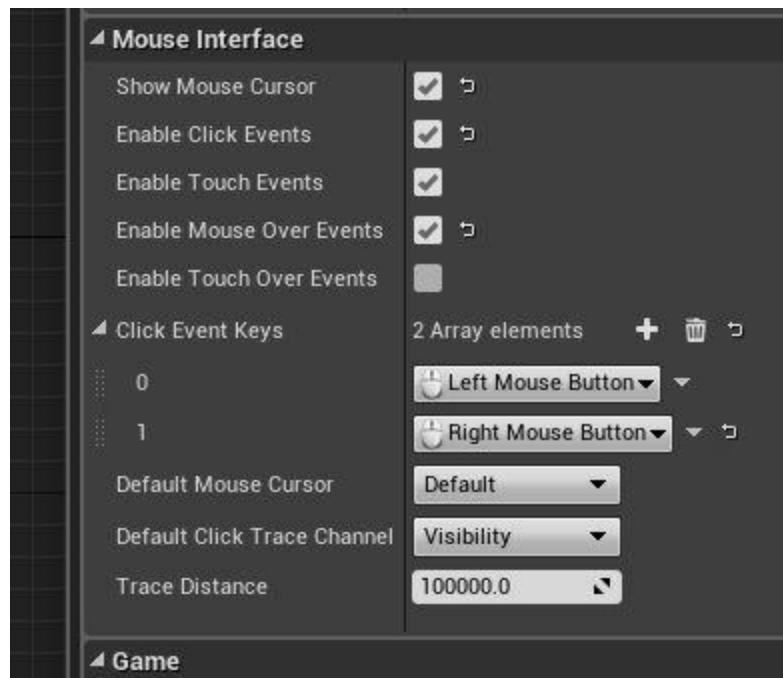
由于使用ActorOnClicked和ActorBeginCursorOver或ActorEndCursorOver事件是自动内  
置到actor蓝图中的，因此，建议您在使用此功能时在游戏中启用鼠标单击/鼠标移过事  
件。



这些事件已经包含在引擎中，并且在任何参与者蓝图的事件图上单击鼠标右键时，可在  
“Add Event”下的“Mouse Input”部分中找到它们：



请注意，必须在播放器控制器的“Mouse Interface”部分中启用“Click Events”或“Mouse Over Events”，这些事件才能在引擎中触发：



## 光标

由于引擎中的错误，此插件不支持自定义鼠标光标。但是，如果您从源代码手动编译，则可以实现以下更改。

若要使用自定义鼠标光标解决此问题，请在引擎源文件的/ Engine / Source / Runtime / WebBrowser / Private / CEF目录中找到CEFWebBrowserWindow.cpp文件。搜索以下所示的方法FCEFWebBrowserWindow::OnCursorChange(...):

```
1695 void FCEFWebBrowserWindow::OnCursorChange(CefCursorHandle CefCursor,
1696 {
1697     switch (Type) {
1698         // Map the basic 3 cursor types directly to Slate types on all platforms.
1699         case CT_NONE:
1700             Cursor = EMouseCursor::None;
1701             break;
1702         case CT_POINTER:
1703             Cursor = EMouseCursor::Default;
1704             break;
1705         case CT_IBEAM:
1706             Cursor = EMouseCursor::TextEditBeam;
1707             break;
1708         #if PLATFORM_WINDOWS || PLATFORM_MAC
1709             // Platform specific support for native cursor types
1710         default:
1711             {
1712                 TSharedPtr PlatformCursor = FSlateApplication::Get().GetPlatformCursor();
1713
1714                 if (PlatformCursor.IsValid())
1715                 {
1716                     PlatformCursor->SetTypeShape(EMouseCursor::Custom);
1717                     Cursor = EMouseCursor::Custom;
1718                 }
1719             }
1720             break;
1721         #else
1722             // Map to closest Slate equivalent on platforms where native cursors are not supported.
1723             case CT_VERTICALTEXT:
1724                 Cursor = EMouseCursor::TextEditBeam;
1725                 break;
1726             }
1727         }
1728     }
1729 }
```

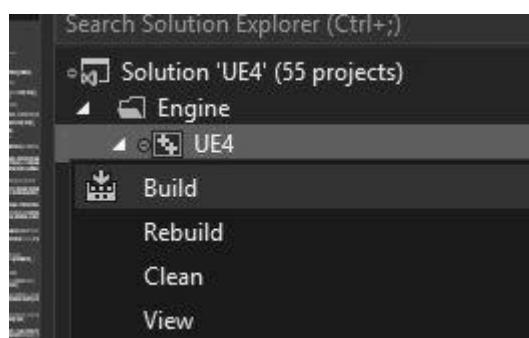
注意第1708行的#if PLATFORM\_WINDOWS || PLATFORM\_MAC。它实质上是通过手动设置操作系统的光标来绕过引擎的本机光标管理的，除非它是默认的指针或文本编辑器光束。

因此，您应将此行更改为#ifndef，以防止编译此块（绝对没有全局设置可切换它）。以下示例说明了对引擎源所做的更改：

```
1705     case CT_IBeam:
1706         Cursor = EMouseCursor::TextEditBeam;
1707         break;
1708 #if 0
1709 // Platform specific support for native cursor types
1710 default:
1711 {
1712     TSharedPtr<ICursor> PlatformCursor = FSlateApplication::Get()
1713
1714     if (PlatformCursor.IsValid())
1715     {
1716         PlatformCursor->SetTypeShape(EMouseCursor::Custom);
1717         Cursor = EMouseCursor::Custom;
1718     }
1719 }
1720 break;
1721 #else
1722 // Map to closest Slate equivalent on platforms where native
1723 case CT_VERTICALEXTENT:
1724     Cursor = EMouseCursor::TextEditBeam;
1725     break;
1726 case CT_EASTRESIZE:
1727 case CT_WESTRESIZE:
1728 case CT_EASTWESTRESIZE:
1729 case CT_COLUMNRESIZE:
1730     Cursor = EMouseCursor::ResizeLeftRight;
1731
break;
```

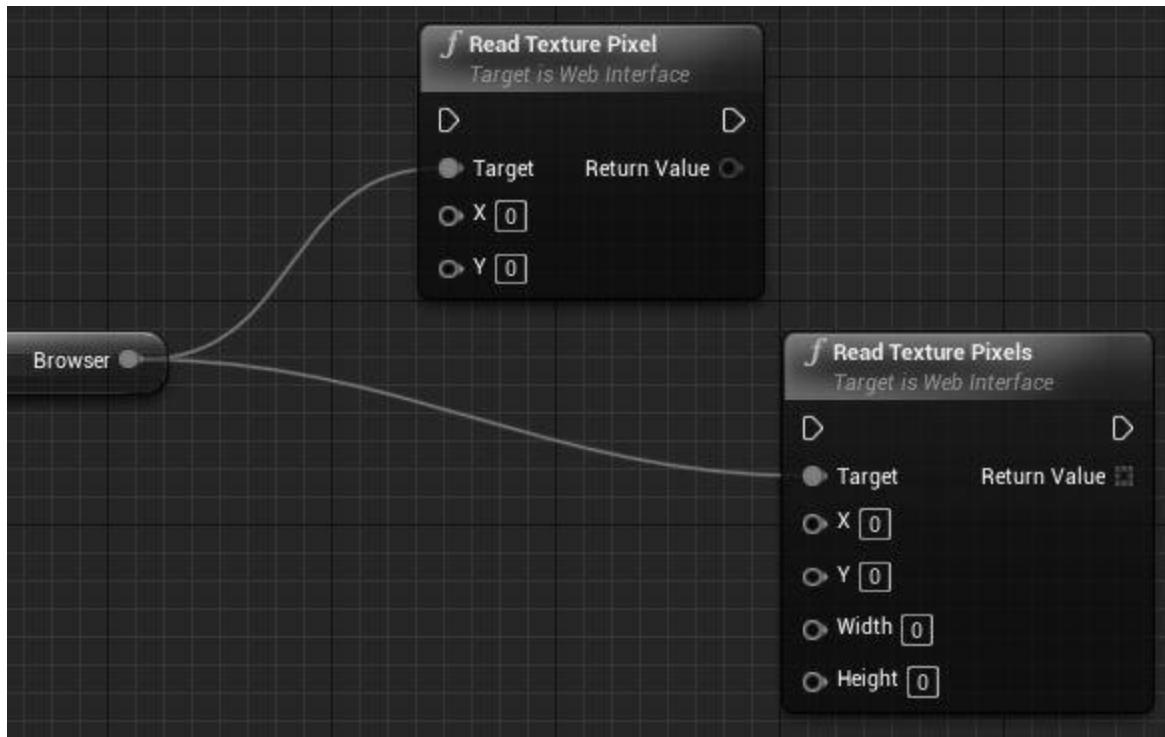
请注意，第1709和1720行之间的块现在显示为灰色，并且Web浏览器进程的光标将被适当地映射到引擎光标类型。

现在，您可以构建UE4项目，以将自定义鼠标光标与Webui插件一起使用：



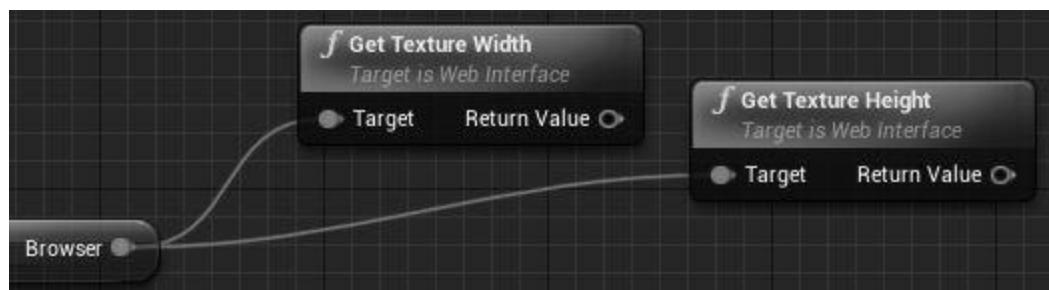
## 纹理

可以使用“Read Texture Pixel”节点或“Read Texture Pixels”节点访问浏览器的纹理数据，如以下示例所示：



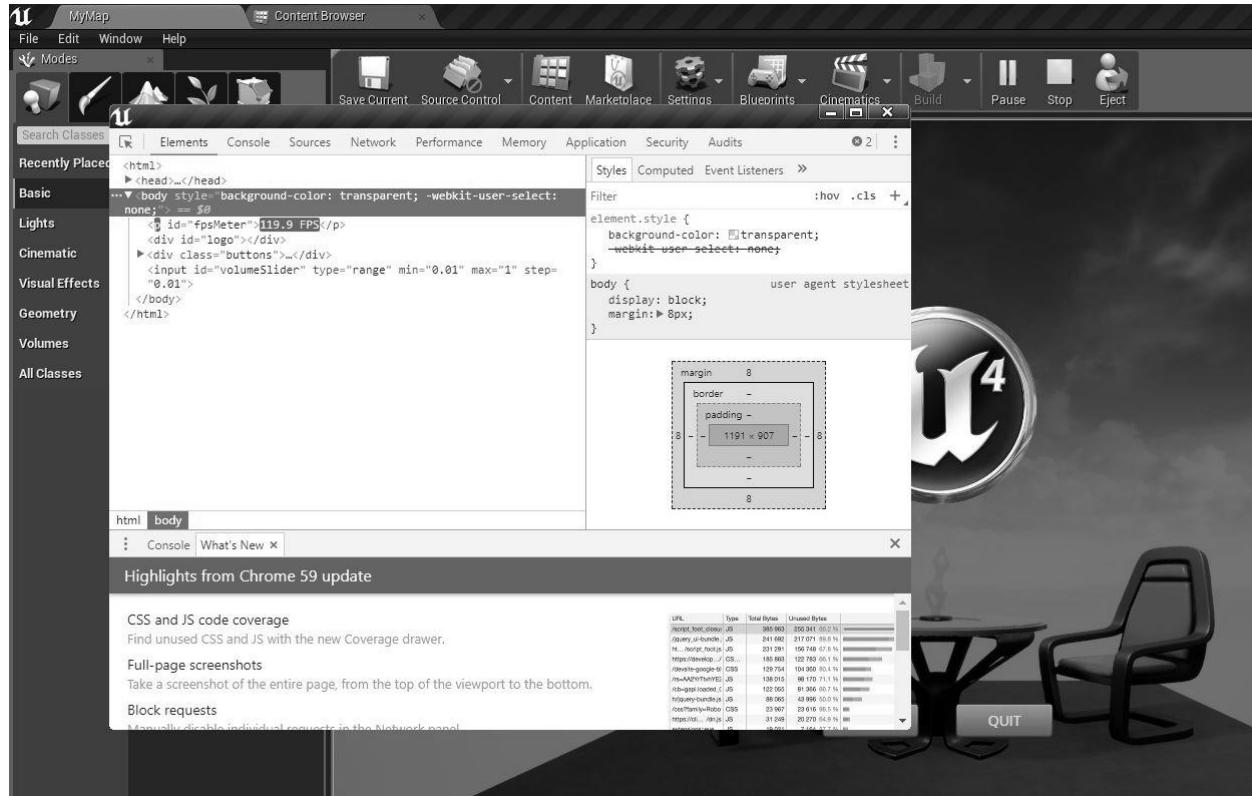
这些节点将分别返回单色或颜色数组。请注意，没有创建任何渲染目标来访问此纹理数据。相反，这些函数使用浏览器的基础纹理引用直接在渲染线程上执行命令。

还请记住，浏览器纹理的大小可能并不总是与视口中的小部件的大小匹配。因此，提供了“Get Texture Width”和“Get Texture Height”节点以访问基础浏览器纹理的大小：



## 工具

聚集对准时，可以使用CTRL + SHIFT + I调试浏览器。此组合键可打开Chrome DevTools，并且仅在开发和调试版本中可用。



您可以在以下地址的Chrome DevTools上找到文档：  
<https://developers.google.com/web/tools/chrome-devtools/>

## WEBGL

由于引擎的配置，此插件在4.23及更低版本的桌面平台上不支持WebGL。但是，如果您从源代码手动进行编译，则可以以性能为代价进行以下更改（尤其是对于全屏界面）。

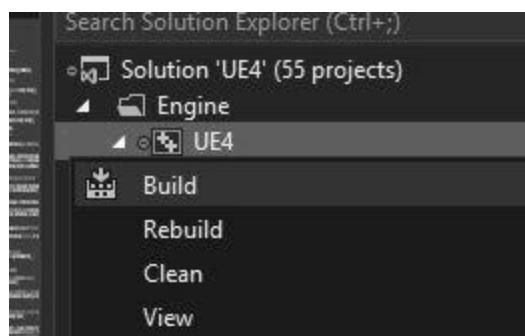
打开/ Engine / Source / Runtime / WebBrowser / Private / CEF目录中的  
CEFBrowserApp.cpp文件。搜索FCEFBrowserApp :: OnBeforeCommandLine  
Processing (...) 方法，如以下示例所示：

```
16     void FCEFBrowserApp::OnBeforeCommandLineProcessing(const CefString& i
17     {
18         CommandLine->AppendSwitch("disable-gpu");
19         CommandLine->AppendSwitch("disable-gpu-compositing");
20 #if !PLATFORM_MAC
21         CommandLine->AppendSwitch("enable-begin-frame-scheduling");
22 #endif
23     }
```

请注意，附加到浏览器子进程命令行的两个开关disable-gpu和disable-gpu-compositing。  
您将需要删除这两行代码，如下所示：

```
16     void FCEFBrowserApp::OnBeforeCommandLineProcessing(const CefString& i
17     {
18 //        CommandLine->AppendSwitch("disable-gpu");
19 //        CommandLine->AppendSwitch("disable-gpu-compositing");
20 #if !PLATFORM_MAC
21         CommandLine->AppendSwitch("enable-begin-frame-scheduling");
22 #endif
23     }
```

现在，您可以构建UE4项目，并在启用WebGL的情况下使用WebUI插件：



## 性能表现

版本4.24到4.26的引擎安装默认情况下启用了WebGL，这会大大降低性能（尤其是对于全屏界面）。但是，如果您从源代码手动编译，则可以实现以下更改。

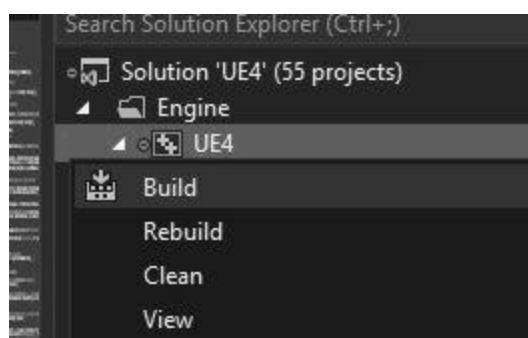
打开/ Engine / Source / Runtime / WebBrowser / Private / CEF目录中的  
CEFBrowserApp.cpp文件。搜索FCEFBrowserApp :: OnBeforeCommandLine Processing  
(...) 方法，如以下示例所示：

```
16     void FCEFBrowserApp::OnBeforeCommandLineProcessing(const CefString& |
17     {
18         CommandLine->AppendSwitch("enable-gpu");
19         CommandLine->AppendSwitch("enable-gpu-compositing");
20         CommandLine->AppendSwitch("enable-begin-frame-scheduling");
21     }
```

请注意，附加到浏览器子进程命令行的两个开关enable-gpu和enable-gpu-compositing。  
您将需要将这两个开关更改为disable-gpu和disable-gpu-compositing，如下所示：

```
16     void FCEFBrowserApp::OnBeforeCommandLineProcessing(const CefString& |
17     {
18         CommandLine->AppendSwitch("disable-gpu");
19         CommandLine->AppendSwitch("disable-gpu-compositing");
20         CommandLine->AppendSwitch("enable-begin-frame-scheduling");
21     }
```

现在，您可以构建UE4项目，以在禁用WebGL的情况下使用Web UI插件：



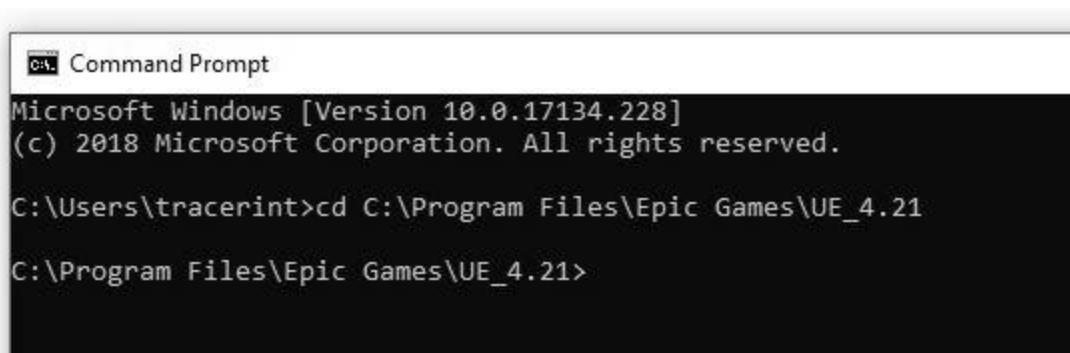
## 编译

可以为其他平台或引擎版本手动编译此插件。首先打开命令提示符（通过在开始菜单中搜索“cmd”）并键入以下命令：

```
cd "C:\Program Files\Epic Games\UE_4.21"
```

您可以通过右键单击命令提示符来复制并粘贴此命令。还要注意UE\_4.21目录。您需要将此文件夹更改为与您所使用的引擎版本相对应的版本。如果未将引擎安装到默认目录，请键入自定义安装文件夹的路径。

按输入运行命令。现在，您应该看到类似于以下内容的输出：



```
Command Prompt
Microsoft Windows [Version 10.0.17134.228]
(c) 2018 Microsoft Corporation. All rights reserved.

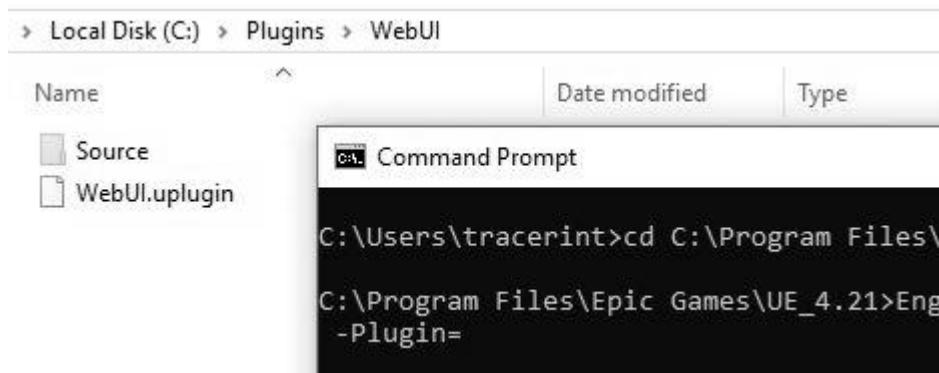
C:\Users\tracerint>cd C:\Program Files\Epic Games\UE_4.21

C:\Program Files\Epic Games\UE_4.21>
```

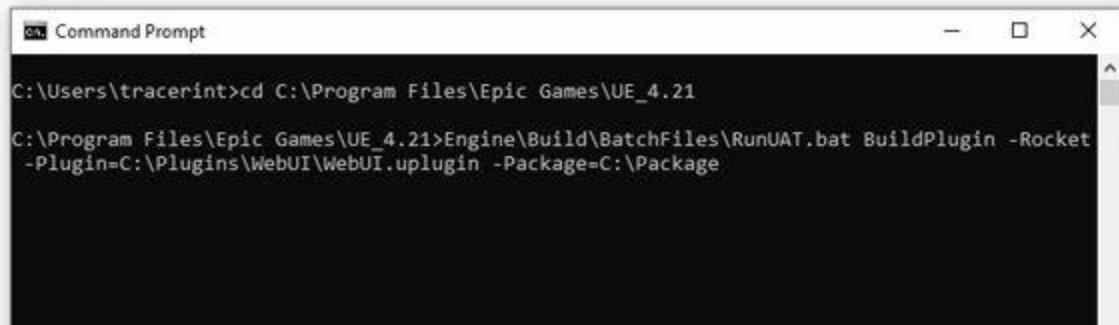
然后键入以下命令来构建插件：

```
Engine\Build\BatchFiles\RunUAT.bat BuildPlugin - Rocket -Plugin="..." -Package="..."
```

将第一个“...”替换为WebUI.uplugin的路径，将最后一个“...”替换为临时“package”文件夹的路径。您也可以将.uplugin文件或您的临时文件夹直接拖放到命令提示符下，它将自动键入路径：



确保这些路径不在引擎目录中，否则构建将失败。输入完整的命令后，其外观应类似于以下内容：



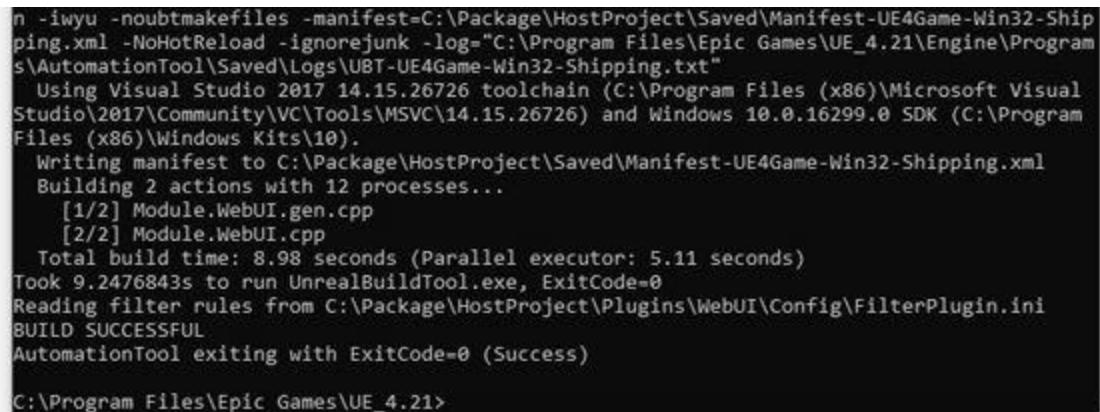
```
cmd Command Prompt
C:\Users\tracerint>cd C:\Program Files\Epic Games\UE_4.21
C:\Program Files\Epic Games\UE_4.21>Engine\Build\BatchFiles\RunUAT.bat BuildPlugin -Rocket
-Plugin=C:\Plugins\WebUI\WebUI.uplugin -Package=C:\Package
```

如果您的计算机不支持Mac或Linux构建，则您很可能必须从WebUI中删除“ Mac”和“ IOS”或“ Linux”平台。 编译前添加：

```
        "Name": "WebUI",
        "Type": "Runtime",
        "LoadingPhase": "PreDefault",
        "WhitelistPlatforms": [
            "Win64",
            "Win32",
            "Linux",
            "Android"
        ]
```

现在按输入键运行命令。 如果一切设置正确，您将看到正在为各种平台编译的插件的许多不同版本。

构建完成后，您应该会看到“ BUILD SUCCESSFUL”信息：



```
n -iwu -noubtmakefiles -manifest=C:\Package\HostProject\Saved\Manifest-UE4Game-Win32-Shipping.xml -NoHotReload -ignorejunk -log="C:\Program Files\Epic Games\UE_4.21\Engine\Programs\AutomationTool\Saved\Logs\UBT-UE4Game-Win32-Shipping.txt"
Using Visual Studio 2017 14.15.26726 toolchain (C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.15.26726) and Windows 10.0.16299.0 SDK (C:\Program Files (x86)\Windows Kits\10).
Writing manifest to C:\Package\HostProject\Saved\Manifest-UE4Game-Win32-Shipping.xml
Building 2 actions with 12 processes...
[1/2] Module.WebUI.gen.cpp
[2/2] Module.WebUI.cpp
Total build time: 8.98 seconds (Parallel executor: 5.11 seconds)
Took 9.2476843s to run UnrealBuildTool.exe, ExitCode=0
Reading filter rules from C:\Package\HostProject\Plugins\WebUI\Config\FilterPlugin.ini
BUILD SUCCESSFUL
AutomationTool exiting with ExitCode=0 (Success)

C:\Program Files\Epic Games\UE_4.21>
```

检查您的临时“package”文件夹，以确保其外观类似于以下内容：

Local Disk (C:) > Package		
Name	Date modified	Type
Binaries	11/29/2018 8:35 AM	File folder
Intermediate	11/29/2018 8:35 AM	File folder
Source	11/29/2018 8:35 AM	File folder
WebUI.uplugin	11/29/2018 8:35 AM	UPLUGIN File

然后，将该临时文件夹中的文件复制到您的引擎安装目录中。如果要编译需要 WebUI 插件的任何其他插件，则必须事先执行此操作。

C:\Users\YourName\Documents\Unreal Engine\Engine\Plugins\Runtime\WebUI		
Name	Date modified	Type
Binaries	11/29/2018 8:35 AM	File folder
Intermediate	11/29/2018 8:35 AM	File folder
Source	11/29/2018 8:35 AM	File folder
WebUI.uplugin	11/29/2018 8:35 AM	UPLUGIN File

您现在已经成功为您的引擎版本编译了 WebUI 插件。