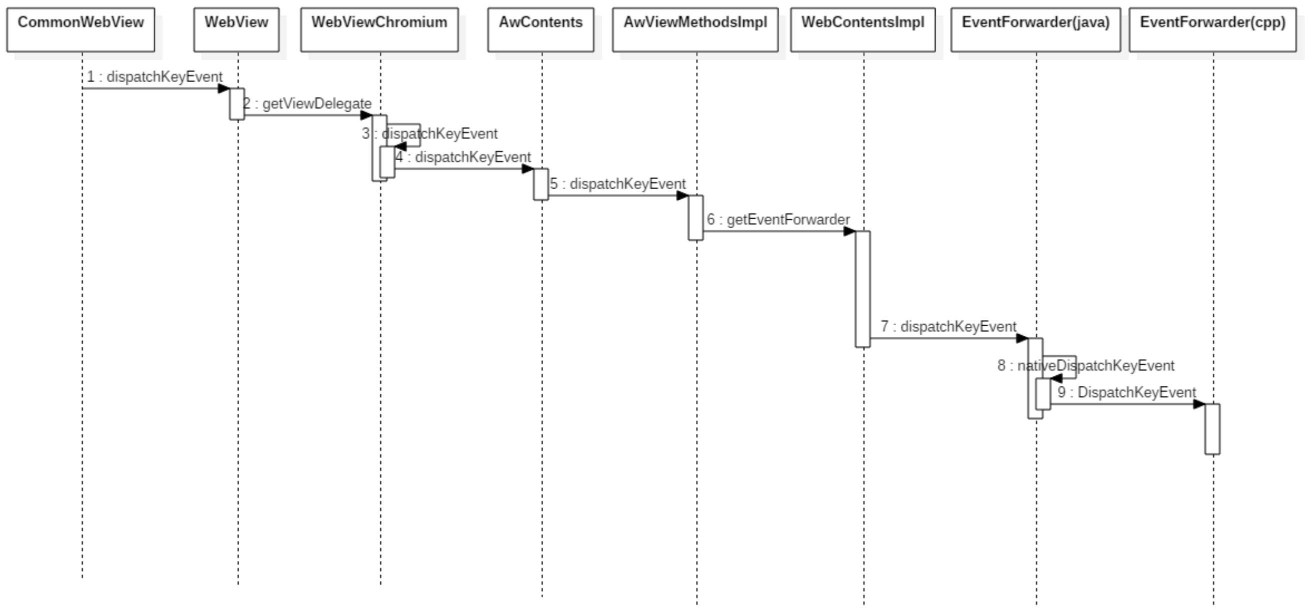
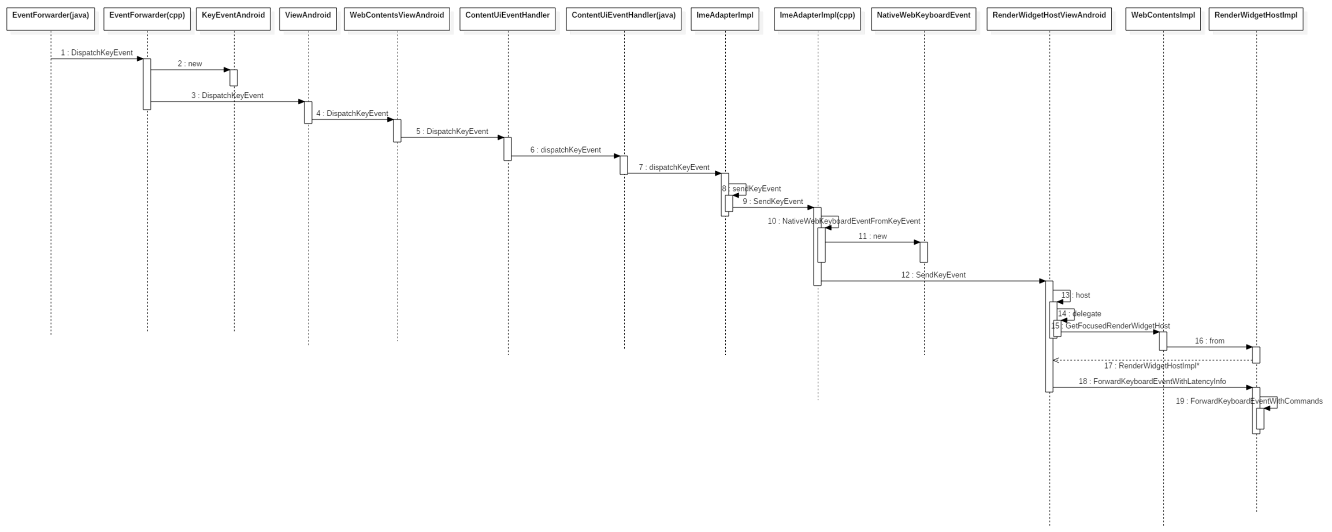
浏览器按键传递流程

本文档介绍浏览器处理按键所经过的所有流程节点。

Java层传递流程时序图：



1. 应用通过WebView的dispatchKeyEvent将按键传递下来，通过WebViewChromium传递到浏览器；
2. 经过几层传递，到EventForwarder，此类会将输入的事件传递到native层；



1. 构造一个KeyEventAndroid对象，该对象用于将java的KeyEvent事件转换为native层的按键事件；
2. 按键处理流程一直传递调用，到ContentUiEventHandler后需要再传回java，这是为了将需要java层处理的UI事件传递回java层处理，主要是处理输入法输入事件；
3. 处理完输入法事件后，再传递回native层继续处理，此处最重要的是需要构造一个NativeWebKeyboardEvent对象，该对象的构造来自于NativeWebKeyboardEvent类，该类根据平台类型有不同的实现，android实现由一个builder生成：

```
WebKeyboardEvent WebKeyboardEventBuilder::Build(
    JNIEnv* env,
    const base::android::JavaRef<jobject>& android_key_event,
    WebInputEvent::Type type,
    int modifiers,
    base::TimeTicks time,
    int keycode,
    int scancode,
    int unicode_character,
    bool is_system_key) {
  DCHECK(WebInputEvent::IsKeyboardEventType(type));

  ui::DomCode dom_code = ui::DomCode::NONE;
  if (scancode)
    dom_code = ui::KeycodeConverter::NativeKeycodeToDomCode(scancode);

  WebKeyboardEvent result(
      type, modifiers | ui::DomCodeToWebInputEventModifiers(dom_code), time);
  result.windows_key_code = ui::LocatedToNonLocatedKeyboardCode(
      ui::KeyboardCodeFromAndroidKeyCode(keycode));
  result.native_key_code = keycode;
  result.dom_code = static_cast<int>(dom_code);
  result.dom_key = GetDomKeyFromEvent(env, android_key_event, keycode,
                                      modifiers, unicode_character);
  result.unmodified_text[0] = unicode_character;
  if (result.windows_key_code == ui::VKEY_RETURN) {
    // This is the same behavior as GTK:
    // We need to treat the enter key as a key press of character \r. This
    // is apparently just how webkit handles it and what it expects.
    result.unmodified_text[0] = '\r';
  }
  result.text[0] = result.unmodified_text[0];
  result.is_system_key = is_system_key;

  return result;
}
```

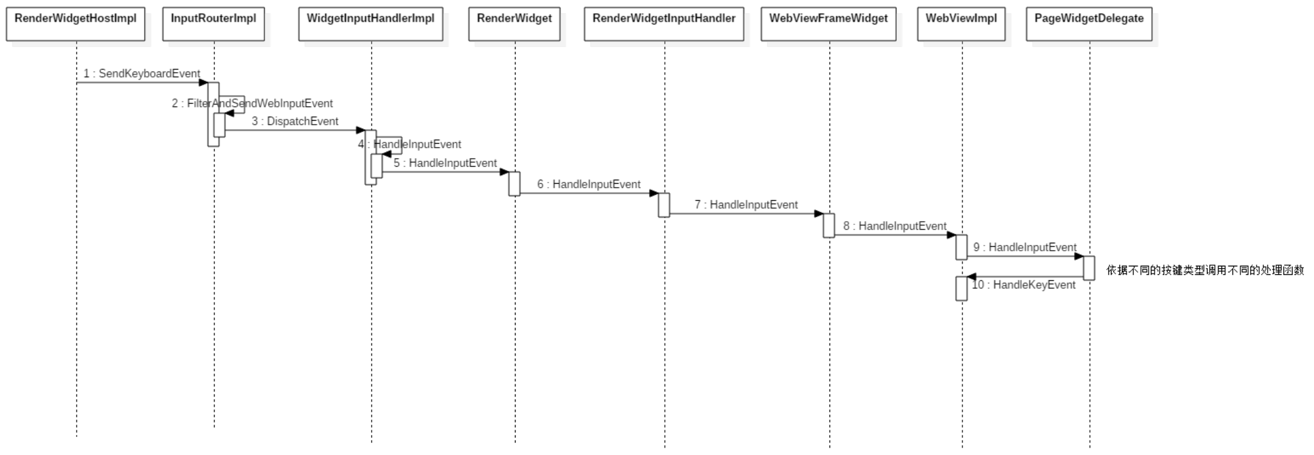
此处构造按键的几个属性：windows_key_code、native_key_code、dom_code、dom_key

windows_key_code来自于keyboard_code_conversion_android.cc文件的KeyboardCodeFromAndroidKeyCode函数转换而来：

```
KeyboardCode KeyboardCodeFromAndroidKeyCode(int keycode) {
  // Does not provide all key codes, and does not handle all keys.
  switch (keycode) {
#if defined(OS_ANDROID)
#define ANDROID_KEYCODE_TO_KB_CODE
#include "ui/events/keycodes/dom/keycode_conversion_data_android_generated.inc"
#undef ANDROID_KEYCODE_TO_KB_CODE
#endif
    case AKEYCODE_DEL:
      return VKEY_BACK;
    case AKEYCODE_TAB:
      return VKEY_TAB;
    case AKEYCODE_CLEAR:
      return VKEY_CLEAR;
    case AKEYCODE_DPAD_CENTER:
    case AKEYCODE_ENTER:
      return VKEY_RETURN;
    case AKEYCODE_SHIFT_LEFT:
      return VKEY_LSHIFT;



    case AKEYCODE_NUMPAD_ADD:
      return VKEY_ADD;
    case AKEYCODE_NUMPAD_DOT:
      return VKEY_DECIMAL;
    case AKEYCODE_CHANNEL_UP:
      return VKEY_PRIOR;
    case AKEYCODE_CHANNEL_DOWN:
      return VKEY_NEXT;
    default:
#if defined(USE_T_EMBEDDED) //add by zhongzw for tcl private key
      return (KeyboardCode)keycode;
#else
      return VKEY_UNKNOWN;
#endif
  }
}
```

dom_key来自于GetDomKeyFromAndroidKeycode函数，经其转换而来

```
DomKey GetDomKeyFromAndroidKeycode(int keycode) {
  switch (keycode) {
    default:
    case AKEYCODE_UNKNOWN:
      return DomKey::NONE;
#if defined(OS_ANDROID)
#define ANDROID_KEYCODE_TO_DOM_KEY
#include "ui/events/keycodes/dom/keycode_conversion_data_android_generated.inc"
#undef ANDROID_KEYCODE_TO_DOM_KEY
#endif
    case AKEYCODE_HOME:
      return DomKey::GO_HOME;
    case AKEYCODE_BACK:
      return DomKey::GO_BACK;
    case AKEYCODE_CALL:
      return DomKey::CALL;
    case AKEYCODE_ENDCALL:
      return DomKey::END_CALL;
    case AKEYCODE_DPAD_UP:
      return DomKey::ARROW_UP;



    case AKEYCODE_COPY:
      return DomKey::COPY;
    case AKEYCODE_PASTE:
      return DomKey::PASTE;
    case AKEYCODE_DVR:
      return DomKey::DVR;
  }
}
```



该按键事件经转换后传给render处理，到WebViewImpl后，判断该按键事件是鼠标还是键盘事件，再去调用对应处理函数