

# function FB variant\_map\_of

BrowserHostPtr  
BrowserHostWrapper  
BrowserObjectAPI  
CallMethodFuncor  
CallMethodPtr  
ConstructType  
convert\_variant\_list  
convert\_variant\_list  
convert\_variant\_list  
convert\_variant\_list  
convert\_variant\_map  
convert\_variant\_map  
convert\_variant\_map  
convert\_variant\_map  
FactoryBasePtr  
FBKeyCode  
GDKKeyCodeToFBKeyCode  
get\_jsapi  
GetPropertyType  
GetPropFuncor  
GetPropPtr  
InvokeType  
JSAPIPtr  
JSAPIWeakPtr  
JSObjectPtr  
JSObjectWeakPtr  
JSOutObject  
make\_property  
make\_property  
make\_variant\_list  
make\_variant\_list  
make\_variant\_list  
make\_variant\_map  
make\_variant\_map  
make\_variant\_map  
make\_variant\_map  
MethodFuncorMap  
MethodMap  
PluginCorePtr  
PropertyFuncorsMap  
PropertyMap  
ptr\_cast  
RemovePropertyType  
SecurityLevel  
SecurityZone  
SetPropertyType  
SetPropFuncor  
SetPropPtr  
StringSet  
utf8\_to\_wstring  
variant\_list\_of  
variant\_list\_of  
variant\_map\_of  
variant\_map\_of  
VariantList  
VariantMap  
WinKeyCodeToFBKeyCode  
wstring\_to\_utf8  
wstring\_tolower

```
template<typename T >
FB::detail::VariantMapInserter< T > FB::variant_map_of ( const T & t,
                                                         const FB::variant & v
                                                         )
```

Allows convenient creation of an **FB::VariantMap**.

Returns

A helper type that overloads operator() for insertion and is convertible to **FB::VariantMap**.

Examples:

```
typedef std::map<std::string, FB::variant> StringVariantMap;
StringVariantMap varMap = FB::variant_map_of<std::string>("a",1)("b",2)("c",
3.4);
FireEvent("randomDiceRoll", FB::variant_map_of<std::string>("foo",42));
std::string x = "xyz";
StringVariantMap varMap2 = FB::variant_map_of(x, 1);
```

Definition at line 134 of file [variant\\_map.h](#).

```
135 {
136 return FB::detail::VariantMapInserter<T>(t, v);
137 }
```

