file ScriptingCore DOM Element.cpp

ScriptingCore/DOM/Element.cpp

```
/**********************************************************\
Original Author: Richard Bateman (taxilian)

Created: Dec 9, 2009
License: Dual license model; choose one of two:
 New BSD License
 http://www.opensource.org/licenses/bsd-license.php
 - or -
 GNU Lesser General Public License, version 2.1
  http://www.gnu.org/licenses/lgpl-2.1.html

Copyright 2009 PacketPass, Inc and the Firebreath development team
\**********************************************************/

#include "variant_list.h"
#include "../precompiled_headers.h" // On windows, everything above this line in PCH
#include "Element.h"

using namespace FB::DOM;

Element::Element(const FB::JSObjectPtr& element) : Node(element)
{
}

Element::~Element()
{
}

std::string Element::getInnerHTML() const
{
 return getProperty<std::string>("innerHTML");
}
void Element::setInnerHTML(const std::string& html) const
{
 setProperty("innerHTML", html);
}

int Element::getWidth() const
{
 return getProperty<int>("width");
}
void Element::setWidth(const int width) const
{
 setProperty("width", width);
}

int Element::getScrollWidth() const
{
 return getProperty<int>("scrollWidth");
}

int Element::getHeight() const
{
 return getProperty<int>("height");
}
void Element::setHeight(const int height) const
{
 setProperty("height", height);
}

int Element::getScrollHeight() const
{
 return getProperty<int>("scrollHeight");
}

int Element::getChildNodeCount() const
{
 return getNode("childNodes")->getProperty<int>("length");
}

ElementPtr Element::getChildNode(const int idx) const
{
```

```
73  ElementPtr retVal(getElement("childNodes")->getElement(idx));
74  return retVal;
75 }
76
77 ElementPtr Element::getParentNode() const
78 {
79  ElementPtr retVal(getElement("parentNode"));
80  return retVal;
81 }
82
83 ElementPtr Element::getElementById(const std::string& id) const
84 {
85  JSObjectPtr api =
86  callMethod<JSObjectPtr>("getElementById", variant_list_of(id));
87  return Element::create(api);
88 }
89
90 std::vector<ElementPtr> Element::getElementsByTagName(const std::wstring& tagName) const
91 {
92  return getElementsByTagName(FB::wstring_to_utf8(tagName));
93 }
94
95 std::vector<ElementPtr> Element::getElementsByTagName(const std::string& tagName) const
96 {
97  std::vector<FB::JSObjectPtr> tagList = callMethod<std::vector<FB::JSObjectPtr> >("getElementsByTagName"
, FB::variant_list_of(tagName));
98  std::vector<FB::JSObjectPtr>::iterator it;
99  std::vector<ElementPtr> outList;
100  for (it = tagList.begin(); it != tagList.end(); ++it)
101  {
102  outList.push_back(Element::create(*it));
103  }
104  return outList;
105 }
106
107 std::string FB::DOM::Element::getStringAttribute( const std::string& attr ) const
108 {
109  return callMethod<std::string>("getAttribute", FB::variant_list_of(attr));
110 }
111
```

FB::DOM::Node
DOM Node wrapper.
Definition: ScriptingCore/DOM/Node.h:39
FB::JSObjectPtr
boost::shared_ptr< FB::JSObject > JSObjectPtr
Defines an alias representing a JSObject shared_ptr (you should never use a JSObject* directly) ...
Definition: APITypes.h:109
FB::variant_list_of
FB::detail::VariantListInserter variant_list_of(FB::variant v)
Allows convenient creation of an FB::VariantList.
Definition: variant_list.h:122
FB::wstring_to_utf8
std::string wstring_to_utf8(const std::wstring &src)
Accepts a std::wstring and returns a UTF8-encoded std::string.
Definition: utf8_tools.cpp:37
FB::DOM::ElementPtr
boost::shared_ptr< Element > ElementPtr
shared_ptr for a FB::DOM::Element
Definition: ScriptingCore/DOM/Element.h:25