ClickBlocks\Utils\DOMDocumentEx

Inheritance	DOMDocument
Subclasses	no
Interfaces	no
Source	Framework/utils/dom.php

DOMDocumentEx is extended operations with the DOM in HTML or XML document.

Public non-static methods

__construct()

```
public void __construct(string $version = '1.0', string $charset = 'utf-8')
```

$version	string	The version number of the document as part of the XML declaration.
$charset	string	The encoding of the document as part of the XML declaration.

Creates a new **DOMDocumentEx** object. The only difference from the **DOMDocument** constructor is the default value of charset is UTF-8, while charset in the parent constructor is not defined.

loadHTML()

```
public boolean loadHTML(string $source, integer $options = 0)
```

$source	string	the HTML string.
$options	integer	bitwise OR of the libxml option constants.

The function parses the HTML contained in the string source. Unlike loading XML, HTML does not have to be well-formed to load. The method returns TRUE on success or FALSE on failure.

> Unlike the parent **loadHTML()** the overridden method does not throw exception during processing malformed HTML.

loadHTMLFile()

```
public boolean loadHTMLFile(string $filename, integer $options = 0)
```

$filename	string	the path to the HTML file.
$options	integer	bitwise OR of the libxml option constants.

The method parses the HTML document in the file named **filename**. Unlike loading XML, HTML does not have to be well-formed to load. The method returns TRUE on success or FALSE on failure.

> Unlike the parent **loadHTMLFile()** the overridden method does not throw exception during processing malformed HTML.

getHTML()

```
public string getHTML(DOMNode $node = null)
```

$node	DOMNode	the given node.

Returns HTML of the given node. The method returns HTML of the root node if **$node** is not defined. Example:

```
$dom = new DOMDocumentEx();
// Creates an HTML document.
$dom->loadHTML('<div id="el">test</div>');
// Displays HTML of the document.
echo $dom->getHTML() . PHP_EOL;
// Gets node object by its ID.
$node = $dom->getElementById('el');
// Displays HTML of the node.
echo $dom->getHTML($node);
```

The above example will output:

```
<html><body><div id="el">test</div></body></html>
<div id="el">test</div>
```

> Note, that in our example after we created HTML document the root node is **html**, even if we have not defined it explicitly. Such behavior is caused by auto-formatting of any HTML documents and it cannot be canceled. When you invoke **loadHTML()** method some tags (such as **!DOCTYPE**, **html**, **body** or **head**) will be automatically added to the document.

setHTML()

```
public DOMNode setHTML(string $html, DOMNode $node = null)
```

$html	string	HTML of the given node.
$node	DOMNode	the given node.

Sets HTML of the given node. If parameter **$node** is not specified HTML of the entire document will be set. This method can also be used for loading HTML document. The method returns object of the given node. Example:

```
$dom = new DOMDocumentEx();
// Loads new HTML document.
$dom->setHTML('<div><b id="el">test</b></div>');
// Displays HTML of the document.
echo $dom->getHTML() . PHP_EOL;
// Gets node object.
$node = $dom->getElementById('el');
// Changes HTML of the node.
$dom->setHTML('<i>test</i>', $node);
// Shows HTML of the node.
echo $dom->getHTML($node);
```

The above example will output:

```
<div><b id="el">test</b></div>
<i>test</i>
```

> Note, that unlike **loadHTML()** as well as **loadHTMLFile()** methods **setHTML()** method does not start auto-formatting of the loading HTML.

getInnerHTML()

```
public string getInnerHTML(DOMNode $node = null)
```

$node	DOMNode	the given node.

Returns the inner HTML of the given node. Example:

```
$dom = new DOMDocumentEx();
$dom->loadHTML('<div id="block"><b>test</b></div>');
echo $dom->getInnerHTML($dom->getElementById('block')); // will output <b>test</b>
```

> If parameter **$node** is not defined **getInnerHTML()** returns inner HTML of the root node.

setInnerHTML()

```
public void setInnerHTML(string $html, DOMNode $node = null)
```

$html	string	the node inner HTML.
$node	DOMNode	the given node.

Sets the inner HTML of the given node. The method returns object of the given node. Example:

```
$dom = new DOMDocumentEx();
$dom->loadHTML('<div id="block"><b>test</b></div>');
$node = $dom->getElementById('block');
$dom->setInnerHTML('<i>test</i>', $node);
echo $dom->getInnerHTML($node); // will output <i>test</i>
```

> if parameter **$node** is not specified **setInnerHTML()** will set inner HTML of the root node.

insert()

```
public DOMNode insert(mixed $id, string $html)
```

$id	string, DOMNode	the node ID.
$html	string	the node inner HTML.

Changes the inner HTML of a node. if the node with the given ID doesn't exist the exception will be thrown. The method returns object of the changed node. Example:

```
$dom = new DOMDocumentEx();
$dom->loadHTML('<div id="block">test</div>');
$node = $dom->insert('block', '<label>test</label>');
echo $dom->getHTML($node); // will output <div id="block"><label>test</label></div>
```

replace()

```
public DOMNode replace(mixed $id, string $html)
```

$id	string, DOMNode	the node ID.
$html	string	HTML for replacing.

Replaces the node HTML by the given HTML. if the node doesn't exist the exception will be thrown. The method returns object of the changed node. Example:

```
$dom = new DOMDocumentEx();
$dom->loadHTML('<div id="block">test</div>');
$node = $dom->replace('block', '<label>test</label>');
echo $dom->getHTML($node); // will output <label>test</label>
```

inject()

```
public void inject(string $id, string $html, string $mode = DOMDocumentEx::DOM_INJECT_TOP)
```

$id	string, DOMNode	the node ID.
$html	string	the HTML for adding.
$mode	string	the injecting mode.

Adds HTML to the node. The optional parameter specifies where the HTML will be added: at the beginning or end, before or after the node. if the node doesn't exist the exception will be thrown. The method returns object of the changed node. Example:

```
$dom = new DOMDocumentEx();
$dom->setHTML('<div id="block"><b>test</b></div>');
$dom->inject('block', '<i>a</i>', 'top');
echo $dom->getHTML() . PHP_EOL;
$dom->inject('block', '<i>b</i>', 'bottom');
echo $dom->getHTML() . PHP_EOL;
$dom->inject('block', '<i>c</i>', 'after');
echo $dom->getHTML() . PHP_EOL;
$dom->inject('block', '<i>d</i>', 'before');
echo $dom->getHTML();
```

The above example will output:

```
<div id="block"><i>a</i><b>test</b></div>
<div id="block"><i>a</i><b>test</b><i>b</i></div>
<div id="block"><i>a</i><b>test</b><i>b</i></div><i>c</i>
<i>d</i><div id="block"><i>a</i><b>test</b><i>b</i></div><i>c</i>
```

HTMLtoNode()

```
public DOMNode HTMLToNode(string $html)
```

$html	string	the HTML for conversion.

Converts the given HTML to the node object. If HTML contains several elements only the first one will be converted to node. Example:

```
$dom = new DOMDocumentEx();
$node = $dom->HTMLtoNode('<div><b>test</b></div>');
echo $dom->getHTML($node); // will output <div><b>test</b></div>
```

getElementById()

```
public DOMElement getElementById(string $id)
```

$id	string	the element unique ID.

Searches and returns an element by its ID. Returns the DOMElement or NULL if the element is not found. Example:

```
$dom = new DOMDocumentEx();
$dom->loadHTML('<div><div id="test"><b>Hello!</b></div></div>');
echo $dom->getHTML($dom->getElementById('test')); // will output <div id="test"><b>Hello!</b></div>
```