SyntaxPlugin.php

```php
<?php

namespace dokuwiki\Extension;

use Doku_Handler;
use Doku_Renderer;

/**
 * Syntax Plugin Prototype
 *
 * All DokuWiki plugins to extend the parser/rendering mechanism
 * need to inherit from this class
 *
 * @license    GPL 2 (http://www.gnu.org/licenses/gpl.html)
 * @author     Andreas Gohr <andi@splitbrain.org>
 */
abstract class SyntaxPlugin extends \dokuwiki\Parsing\ParserMode\Plugin
{
    use PluginTrait;

    protected $allowedModesSetup = false;

    /**
     * Syntax Type
     *
     * Needs to return one of the mode types defined in $PARSER_MODES in Parser.php
     *
     * @return string
     */
    abstract public function getType();

    /**
     * Allowed Mode Types
     *
     * Defines the mode types for other dokuwiki markup that maybe nested within the
     * plugin's own markup. Needs to return an array of one or more of the mode types
     * defined in $PARSER_MODES in Parser.php
     *
     * @return array
     */
    public function getAllowedTypes()
    {
        return array();
    }
```

```
45.
46.     /**
47.      * Paragraph Type
48.      *
49.      * Defines how this syntax is handled regarding paragraphs.
    This is important
50.      * for correct XHTML nesting. Should return one of the
    following:
51.      *
52.      * 'normal' - The plugin can be used inside paragraphs
53.      * 'block'  - Open paragraphs need to be closed before plugin
    output
54.      * 'stack'  - Special case. Plugin wraps other paragraphs.
55.      *
56.      * @see Doku_Handler_Block
57.      *
58.      * @return string
59.      */
60.     public function getPType()
61.     {
62.         return 'normal';
63.     }
64.
65.     /**
66.      * Handler to prepare matched data for the rendering process
67.      *
68.      * This function can only pass data to render() via its return
    value - render()
69.      * may be not be run during the object's current life.
70.      *
71.      * Usually you should only need the $match param.
72.      *
73.      * @param   string $match The text matched by the patterns
74.      * @param   int $state The lexer state for the match
75.      * @param   int $pos The character position of the matched
    text
76.      * @param   Doku_Handler $handler The Doku_Handler object
77.      * @return  bool|array Return an array with all data you want
    to use in render, false don't add an instruction
78.      */
79.     abstract public function handle($match, $state, $pos,
    Doku_Handler $handler);
80.
81.     /**
82.      * Handles the actual output creation.
83.      *
84.      * The function must not assume any other of the classes
    methods have been run
85.      * during the object's current life. The only reliable data it
```

```
    receives are its
 86.      * parameters.
 87.      *
 88.      * The function should always check for the given output
    format and return false
 89.      * when a format isn't supported.
 90.      *
 91.      * $renderer contains a reference to the renderer object which
    is
 92.      * currently handling the rendering. You need to use it for
    writing
 93.      * the output. How this is done depends on the renderer used
    (specified
 94.      * by $format
 95.      *
 96.      * The contents of the $data array depends on what the
    handler() function above
 97.      * created
 98.      *
 99.      * @param string $format output format being rendered
100.      * @param Doku_Renderer $renderer the current renderer object
101.      * @param array $data data created by handler()
102.      * @return  boolean                 rendered correctly?
    (however, returned value is not used at the moment)
103.      */
104.     abstract public function render($format, Doku_Renderer
    $renderer, $data);
105.
106.     /**
107.      *  There should be no need to override this function
108.      *
109.      * @param string $mode
110.      * @return bool
111.      */
112.     public function accepts($mode)
113.     {
114.
115.         if (!$this->allowedModesSetup) {
116.             global $PARSER_MODES;
117.
118.             $allowedModeTypes = $this->getAllowedTypes();
119.             foreach ($allowedModeTypes as $mt) {
120.                 $this->allowedModes =
    array_merge($this->allowedModes, $PARSER_MODES[$mt]);
121.             }
122.
123.             $idx = array_search(substr(get_class($this), 7),
    (array)$this->allowedModes);
124.             if ($idx !== false) {
125.                 unset($this->allowedModes[$idx]);
126.             }
```

```
127.             $this->allowedModesSetup = true;
128.         }
129.
130.         return parent::accepts($mode);
131.     }
132. }
```