```php
<?php

/**
 * Функции шаблонов DokuWiki
 *
 * @license    GPL 2 (http://www.gnu.org/licenses/gpl.html)
 * @author     Andreas Gohr <andi@splitbrain.org>
 */

use dokuwiki\ActionRouter;
use dokuwiki\Action\Exception\FatalException;
use dokuwiki\Extension\PluginInterface;
use dokuwiki\Ui\Admin;
use dokuwiki\StyleUtils;
use dokuwiki\Menu\Item\AbstractItem;
use dokuwiki\Form\Form;
use dokuwiki\Menu\MobileMenu;
use dokuwiki\Ui\Subscribe;
use dokuwiki\Extension\AdminPlugin;
use dokuwiki\Extension\Event;
use dokuwiki\File\PageResolver;

/**
* Доступ к файлу шаблона
*
* Возвращает путь к указанному файлу внутри текущего шаблона, использует
* шаблон по умолчанию, если пользовательская версия не существует.
*
* @param  string $ файл
* @возвращаемая строка
*
* @автор Андреас Гор <andi@splitbrain.org>
*/
function template($file)
{
    global $conf;

    if (@is_readable(DOKU_INC . 'lib/tpl/' . $conf['template'] . '/' . $file))
        return DOKU_INC . 'lib/tpl/' . $conf['template'] . '/' . $file;

    return DOKU_INC . 'lib/tpl/dokuwiki/' . $file;
}

/**
* Удобная функция для доступа к каталогу шаблонов из локальной ФС
*
* Заменяет устаревшую константу DOKU_TPLINC.
*
* @param  string $ tpl Шаблон для использования, по умолчанию текущий
* @возвращаемая строка
```

```
51*
52* @автор Андреас Гор <andi@splitbrain.org>
53*/
54 function tpl_incdir($tpl = '')
55 {
56     global $conf;
57     if (!$tpl) $tpl = $conf['template'];
58     return DOKU_INC . 'lib/tpl/' . $tpl . '/';
59 }
60
61/**
62* Удобная функция доступа к каталогу шаблонов из Интернета
63*
64* Заменяет устаревшую константу DOKU_TPL
65*
66* @param  string $ tpl Шаблон для использования, по умолчанию текущий
67* @возвращаемая строка
68*
69* @автор Андреас Гор <andi@splitbrain.org>
70*/
71 function tpl_basedir($tpl = '')
72 {
73     global $conf;
74     if (!$tpl) $tpl = $conf['template'];
75     return DOKU_BASE . 'lib/tpl/' . $tpl . '/';
76 }
77
78/**
79* Распечатать содержимое
80*
81* Эта функция используется для печати всего обычного контента.
82* (определяется глобальной переменной $ACT) путем вызова соответствующего
83* выходные функции из html.php
84*
85* Все, что не использует основной файл шаблона, не
86* обрабатывается этой функцией. ACL-списки здесь тоже не обрабатываются.
87*
88* @param  bool $ prependTOC следует ли здесь отображать оглавление?
89* @return  bool true, если есть какой-либо вывод
90*
91* @triggers TPL_ACT_RENDER
92* @triggers TPL_CONTENT_DISPLAY
93* @автор Андреас Гор <andi@splitbrain.org>
94*/
95 function tpl_content($prependTOC = true)
96 {
97     global $ACT;
98     global $INFO;
99     $INFO['prependTOC'] = $prependTOC;
100
101      ob_start();
```

```
102     Event::createAndTrigger('TPL_ACT_RENDER', $ACT, 'tpl_content_core');
103     $html_output = ob_get_clean();
104     Event::createAndTrigger('TPL_CONTENT_DISPLAY', $html_output,
function ($html_output) {
105         echo $html_output;
106     });
107
108     return !empty($html_output);
109 }
110
111/**
112* Действие по умолчанию  TPL_ACT_RENDER
113*
114* @return  bool
115*/
116 function tpl_content_core()
117 {
118     $router = ActionRouter::getInstance();
119     try {
120         $router->getAction()->tplContent();
121     } catch (FatalException $e) {
122         // there was no content for the action
123         msg(hsc($e->getMessage()), -1);
124         return false;
125     }
126     return true;
127 }
128
129/**
130* Размещает оглавление там, где вызывается функция
131*
132* Если вы используете это, вы, скорее всего, захотите вызвать  tpl_content  с помощью
133* ложный аргумент
134*
135* @param  bool $ return  Следует ли вернуть оглавление вместо его печати?
136* @возвращаемая  строка
137*
138* @автор Андреас Гор <andi@splitbrain.org>
139*/
140 function tpl_toc($return = false)
141 {
142     global $TOC;
143     global $ACT;
144     global $ID;
145     global $REV;
146     global $INFO;
147     global $conf;
148     $toc = [];
149
150     if (is_array($TOC)) {
151         // if a TOC was prepared in global scope, always use it
```

```
152         $toc = $TOC;
153     } elseif (($ACT == 'show' || str_starts_with($ACT, 'export')) &&
!$REV && $INFO['exists']) {
154         // get TOC from metadata, render if neccessary
155         $meta = p_get_metadata($ID, '', METADATA_RENDER_USING_CACHE);
156         $tocok = $meta['internal']['toc'] ?? true;
157         $toc = $meta['description']['tableofcontents'] ?? null;
158         if (!$tocok || !is_array($toc) || !$conf['tocminheads'] ||
count($toc) < $conf['tocminheads']) {
159             $toc = [];
160         }
161     } elseif ($ACT == 'admin') {
162         // try to load admin plugin TOC
163         /** @var AdminPlugin $plugin */
164         if ($plugin = plugin_getRequestAdminPlugin()) {
165             $toc = $plugin->getTOC();
166             $TOC = $toc; // avoid later rebuild
167         }
168     }
169
170     Event::createAndTrigger('TPL_TOC_RENDER', $toc, null, false);
171     $html = html_TOC($toc);
172     if ($return) return $html;
173     echo $html;
174     return '';
175 }
176
177/**
178* Обработка содержимого страницы администратора
179*
180* @return  bool
181*
182* @автор Андреас Гор <andi@splitbrain.org>
183*/
184 function tpl_admin()
185 {
186     global $INFO;
187     global $TOC;
188     global $INPUT;
189
190     $plugin = null;
191     $class = $INPUT->str('page');
192     if (!empty($class)) {
193         $pluginlist = plugin_list('admin');
194
195         if (in_array($class, $pluginlist)) {
196             // attempt to load the plugin
197             /** @var AdminPlugin $plugin */
198             $plugin = plugin_load('admin', $class);
199         }
200     }
```

```
201
202     if ($plugin instanceof PluginInterface) {
203         if (!is_array($TOC)) $TOC = $plugin->getTOC(); //if TOC wasn't
requested yet
204         if ($INFO['prependTOC']) tpl_toc();
205         $plugin->html();
206     } else {
207         $admin = new Admin();
208         $admin->show();
209     }
210     return true;
211 }
212
213/**
214* Распечатайте правильные HTML-мета-заголовки
215*
216* Это необходимо разместить в заголовке вашего шаблона.
217*
218* @param  bool $ alt Нужно ли добавлять каналы и ссылки альтернативного
формата?
219* @return  bool
220* @вызывает  JsonException
221*
222* @автор Андреас Гор <andi@splitbrain.org>
223* @triggers TPL_METAHEADER_OUTPUT
224*/
225 function tpl_metaheaders($alt = true)
226 {
227     global $ID;
228     global $REV;
229     global $INFO;
230     global $JSINFO;
231     global $ACT;
232     global $QUERY;
233     global $lang;
234     global $conf;
235     global $updateVersion;
236     /** @var Input $INPUT */
237     global $INPUT;
238
239    // подготавливаем массив головок
240     $head = [];
241
242    // подготовить seed для js и css
243     $tseed = $updateVersion;
244     $depends = getConfigFiles('main');
245     $depends[] = DOKU_CONF . "tpl/" . $conf['template'] . "/style.ini";
246     foreach ($depends as $f) $tseed .= @filemtime($f);
247     $tseed = md5($tseed);
248
249    // обычные вещи
```

```
250     $head['meta'][] = ['name' => 'generator', 'content' => 'DokuWiki'];
251     if (actionOK('search')) {
252         $head['link'][] = [
253             'rel' => 'search',
254             'type' => 'application/opensearchdescription+xml',
255             'href' => DOKU_BASE . 'lib/exe/opensearch.php',
256             'title' => $conf['title']
257         ];
258     }
259 
260     $head['link'][] = ['rel' => 'start', 'href' => DOKU_BASE];
261     if (actionOK('index')) {
262         $head['link'][] = [
263             'rel' => 'contents',
264             'href' => wl($ID, 'do=index', false, '&'),
265             'title' => $lang['btn_index']
266         ];
267     }
268 
269     if (actionOK('manifest')) {
270         $head['link'][] = [
271             'rel' => 'manifest',
272             'href' => DOKU_BASE . 'lib/exe/manifest.php'
273         ];
274     }
275 
276     $styleUtil = new StyleUtils();
277     $styleIni = $styleUtil->cssStyleini();
278     $replacements = $styleIni['replacements'];
279     if (!empty($replacements['__theme_color__'])) {
280         $head['meta'][] = [
281             'name' => 'theme-color',
282             'content' => $replacements['__theme_color__']
283         ];
284     }
285 
286     if ($alt) {
287         if (actionOK('rss')) {
288             $head['link'][] = [
289                 'rel' => 'alternate',
290                 'type' => 'application/rss+xml',
291                 'title' => $lang['btn_recent'],
292                 'href' => DOKU_BASE . 'feed.php'
293             ];
294             $head['link'][] = [
295                 'rel' => 'alternate',
296                 'type' => 'application/rss+xml',
297                 'title' => $lang['currentns'],
298                 'href' => DOKU_BASE . 'feed.php?mode=list&ns=' .
(isset($INFO) ? $INFO['namespace'] : '')
299             ];
```

```
300         }
301         if (($ACT == 'show' || $ACT == 'search') && $INFO['writable']) {
302             $head['link'][] = [
303                 'rel' => 'edit',
304                 'title' => $lang['btn_edit'],
305                 'href' => wl($ID, 'do=edit', false, '&')
306             ];
307         }
308 
309         if (actionOK('rss') && $ACT == 'search') {
310             $head['link'][] = [
311                 'rel' => 'alternate',
312                 'type' => 'application/rss+xml',
313                 'title' => $lang['searchresult'],
314                 'href' => DOKU_BASE . 'feed.php?mode=search&q=' . $QUERY
315             ];
316         }
317 
318         if (actionOK('export_xhtml')) {
319             $head['link'][] = [
320                 'rel' => 'alternate',
321                 'type' => 'text/html',
322                 'title' => $lang['plainhtml'],
323                 'href' => exportlink($ID, 'xhtml', '', false, '&')
324             ];
325         }
326 
327         if (actionOK('export_raw')) {
328             $head['link'][] = [
329                 'rel' => 'alternate',
330                 'type' => 'text/plain',
331                 'title' => $lang['wikimarkup'],
332                 'href' => exportlink($ID, 'raw', '', false, '&')
333             ];
334         }
335     }
336 
337     // настройка тегов робота, подходящих для разных режимов
338     if (($ACT == 'show' || $ACT == 'export_xhtml') && !$REV) {
339         if ($INFO['exists']) {
340             //delay indexing:
341             if ((time() - $INFO['lastmod']) >= $conf['indexdelay'] &&
!isHiddenPage($ID)) {
342                 $head['meta'][] = ['name' => 'robots', 'content' =>
'index,follow'];
343             } else {
344                 $head['meta'][] = ['name' => 'robots', 'content' =>
'noindex,nofollow'];
345             }
346             $canonicalUrl = wl($ID, '', true, '&');
347             if ($ID == $conf['start']) {
```

```
348                 $canonicalUrl = DOKU_URL;
349             }
350             $head['link'][] = ['rel' => 'canonical', 'href' =>
$canonicalUrl];
351         } else {
352             $head['meta'][] = ['name' => 'robots', 'content' =>
'noindex,follow'];
353         }
354     } elseif (defined('DOKU_MEDIADETAIL')) {
355         $head['meta'][] = ['name' => 'robots', 'content' =>
'index,follow'];
356     } else {
357         $head['meta'][] = ['name' => 'robots', 'content' =>
'noindex,nofollow'];
358     }
359
360     // установить метаданные
361     if ($ACT == 'show' || $ACT == 'export_xhtml') {
362         // keywords (explicit or implicit)
363         if (!empty($INFO['meta']['subject'])) {
364             $head['meta'][] = ['name' => 'keywords', 'content' =>
implode(',', $INFO['meta']['subject'])];
365         } else {
366             $head['meta'][] = ['name' => 'keywords', 'content' =>
str_replace(':', ',', $ID)];
367         }
368     }
369
370     // загрузка таблиц стилей
371     $head['link'][] = [
372         'rel' => 'stylesheet',
373         'href' => DOKU_BASE . 'lib/exe/css.php?t=' .
rawurlencode($conf['template']) . '&tseed=' . $tseed
374     ];
375
376     $script = "var NS='" . (isset($INFO) ? $INFO['namespace'] : '') .
"';";
377     if ($conf['useacl'] && $INPUT->server->str('REMOTE_USER')) {
378         $script .= "var SIG=" . toolbar_signature() . ";";
379     }
380     jsinfo();
381     $script .= 'var JSINFO = ' . json_encode($JSINFO,
JSON_THROW_ON_ERROR) . ';';
382     $script .= '(function(H){H.className=H.className.replace(/\bno-
js\b/,\'js\')})(document.documentElement);';
383     $head['script'][] = ['_data' => $script];
384
385     // загрузить jquery
386     $jquery = getCdnUrls();
387     foreach ($jquery as $src) {
388         $head['script'][] = [
```

```
389                 '_data' => '',
390                 'src' => $src
391             ] + ($conf['defer_js'] ? ['defer' => 'defer'] : []);
392     }
393
394     // загружаем наш диспетчер javascript
395     $head['script'][] = [
396             '_data' => '',
397             'src' => DOKU_BASE . 'lib/exe/js.php' . '?t=' .
rawurlencode($conf['template']) . '&tseed=' . $tseed
398         ] + ($conf['defer_js'] ? ['defer' => 'defer'] : []);
399
400     // вызвать событие здесь
401     Event::createAndTrigger('TPL_METAHEADER_OUTPUT', $head,
'_tpl_metaheaders_action', true);
402     return true;
403 }
404
405/**
406* печатает массив, созданный tpl_metaheaders
407*
408* $data — это массив различных тегов заголовков. Каждый тег может иметь несколько
409* экземпляры. Атрибуты задаются как пары ключ-значение. Значения будут HTML
410* кодируются автоматически, поэтому их следует предоставлять как есть в массиве
$data.
411*
412* Для тегов, имеющих атрибут body, укажите данные body в специальном поле
413* атрибут '_data'. Это поле НЕ БУДЕТ ЭКРАНИРОВАНО автоматически.
414*
415* Встроенные скрипты будут использовать любой одноразовый номер, указанный в
переменной среды «NONCE».
416*
417* @param   массив $ данные
418*
419* @автор Андреас Гор <andi@splitbrain.org>
420*/
421 function _tpl_metaheaders_action($data)
422 {
423     $nonce = getenv('NONCE');
424     foreach ($data as $tag => $inst) {
425         foreach ($inst as $attr) {
426             if (empty($attr)) {
427                 continue;
428             }
429             if ($nonce && $tag == 'script' && !empty($attr['_data'])) {
430                 $attr['nonce'] = $nonce; // add nonce to inline script
tags
431             }
432             echo '<', $tag, ' ', buildAttributes($attr);
433             if (isset($attr['_data']) || $tag == 'script') {
434                 echo '>', $attr['_data'] ?? '', '</', $tag, '>';
```

```
435             } else {
436                 echo '/>';
437             }
438             echo "\n";
439         }
440     }
441 }
442
443/**
444* Вывести данный скрипт как встроенный тег скрипта
445*
446* Эта функция добавит атрибут nonce, если он доступен.
447*
448* Скрипт НЕ экранируется автоматически!
449*
450* @param  string $ скрипт
451* @param  bool $ return Возврат или прямая печать?
452* @return  string | недействительный
453*/
454 function tpl_inlineScript($script, $return = false)
455 {
456     $nonce = getenv('NONCE');
457     if ($nonce) {
458         $script = '<script nonce="' . $nonce . '">' . $script .
'</script>';
459     } else {
460         $script = '<script>' . $script . '</script>';
461     }
462
463     if ($return) return $script;
464     echo $script;
465 }
466
467/**
468* Распечатать ссылку
469*
470* Просто создает ссылку.
471*
472* @param  string $ url
473* @param  string $ имя
474* @param  string $ еще
475* @param  bool $ return если true вернуть ссылку html, в противном случае
вывести
476* @return  bool | строка html ссылки или true, если выводится
477*
478* @автор Андреас Гор <andi@splitbrain.org>
479*/
480 function tpl_link($url, $name, $more = '', $return = false)
481 {
482     $out = '<a href="' . $url . '" ';
483     if ($more) $out .= ' ' . $more;
```

```php
    $out .= ">$name</a>";
    if ($return) return $out;
    echo $out;
    return true;
}

/**
* Печатает ссылку на WikiPage
*
* Обертка вокруг html_wikilink
*
* @param  string $ id идентификатор страницы
* @param  string | null $ name имя ссылки
* @param  bool $ возврат
* @return  true | строка
*
* @автор Андреас Гор <andi@splitbrain.org>
*/
function tpl_pagelink($id, $name = null, $return = false)
{
    $out = '<bdi>' . html_wikilink($id, $name) . '</bdi>';
    if ($return) return $out;
    echo $out;
    return true;
}

/**
* получить родительскую страницу
*
* Пытается выяснить, какая страница является родительской.
* возвращает false, если ничего не доступно
*
* @param  string $ id идентификатор страницы
* @return  false | строка
*
* @автор Андреас Гор <andi@splitbrain.org>
*/
function tpl_getparent($id)
{
    $resolver = new PageResolver('root');

    $parent = getNS($id) . ':';
    $parent = $resolver->resolveId($parent);
    if ($parent == $id) {
        $pos = strrpos(getNS($id), ':');
        $parent = substr($parent, 0, $pos) . ':';
        $parent = $resolver->resolveId($parent);
        if ($parent == $id) return false;
    }
    return $parent;
}
```

```
535
536/**
537* Распечатать одну из кнопок
538*
539* @param  string $ тип
540* @param  bool $ возврат
541* @return  bool | string html, или false, если данных нет, true, если
выведено
542* @see     tpl_get_action
543*
544* @автор Адриан Лэнг <mail@adrianlang.de>
545* @deprecated 2017-09-01 см. devel:menus
546*/
547 function tpl_button($type, $return = false)
548 {
549     dbg_deprecated('see devel:menus');
550     $data = tpl_get_action($type);
551     if ($data === false) {
552         return false;
553     } elseif (!is_array($data)) {
554         $out = sprintf($data, 'button');
555     } else {
556         /**
557          * @var string $accesskey
558          * @var string $id
559          * @var string $method
560          * @var array $params
561          */
562         extract($data);
563         if ($id === '#dokuwiki__top') {
564             $out = html_topbtn();
565         } else {
566             $out = html_btn($type, $id, $accesskey, $params, $method);
567         }
568     }
569     if ($return) return $out;
570     echo $out;
571     return true;
572 }
573
574/**
575* Как кнопки действий, но ссылки
576*
577* @param  string $ тип действие команда
578* @param  string $ pre префикс ссылки
579* @param  string $ suf суффикс ссылки
580* @param  string $ внутренний innerHML ссылки
581* @param  bool $ return если true, то возвращает html, в противном случае
печатает
582* @return  bool | string html или false, если данных нет, true, если
выведено
```

```
583*
584* @see     tpl_get_action
585* @автор Адриан Лэнг <mail@adrianlang.de>
586* @deprecated 2017-09-01 см. devel:menus
587*/
588 function tpl_actionlink($type, $pre = '', $suf = '', $inner = '',
$return = false)
589 {
590     dbg_deprecated('see devel:menus');
591     global $lang;
592     $data = tpl_get_action($type);
593     if ($data === false) {
594         return false;
595     } elseif (!is_array($data)) {
596         $out = sprintf($data, 'link');
597     } else {
598         /**
599          * @var string $accesskey
600          * @var string $id
601          * @var string $method
602          * @var bool $nofollow
603          * @var array $params
604          * @var string $replacement
605          */
606         extract($data);
607         if (strpos($id, '#') === 0) {
608             $linktarget = $id;
609         } else {
610             $linktarget = wl($id, $params);
611         }
612         $caption = $lang['btn_' . $type];
613         if (strpos($caption, '%s')) {
614             $caption = sprintf($caption, $replacement);
615         }
616         $akey = '';
617         $addTitle = '';
618         if ($accesskey) {
619             $akey = 'accesskey="' . $accesskey . '" ';
620             $addTitle = ' [' . strtoupper($accesskey) . ']';
621         }
622         $rel = $nofollow ? 'rel="nofollow" ' : '';
623         $out = tpl_link(
624             $linktarget,
625             $pre . ($inner ?: $caption) . $suf,
626             'class="action ' . $type . '" ' .
627             $akey . $rel .
628             'title="' . hsc($caption) . $addTitle . '"',
629             true
630         );
631     }
632     if ($return) return $out;
```

```
633     echo $out;
634     return true;
635 }
636 
637/**
638* Проверьте действия и получите данные для кнопок и ссылок
639*
640* @param  string $ тип
641* @return  массив | bool | строка
642*
643* @автор Адриан Лэнг <mail@adrianlang.de>
644* @автор Андреас Гор <andi@splitbrain.org>
645* @автор Маттиас Гримм <matthiasgrimm@users.sourceforge.net>
646* @deprecated 2017-09-01 см. devel:menus
647*/
648 function tpl_get_action($type)
649 {
650     dbg_deprecated('see devel:menus');
651     if ($type == 'history') $type = 'revisions';
652     if ($type == 'subscription') $type = 'subscribe';
653     if ($type == 'img_backto') $type = 'imgBackto';
654 
655     $class = '\\dokuwiki\\Menu\\Item\\' . ucfirst($type);
656     if (class_exists($class)) {
657         try {
658             /** @var AbstractItem $item */
659             $item = new $class();
660             $data = $item->getLegacyData();
661             $unknown = false;
662         } catch (RuntimeException $ignored) {
663             return false;
664         }
665     } else {
666         global $ID;
667         $data = [
668             'accesskey' => null,
669             'type' => $type,
670             'id' => $ID,
671             'method' => 'get',
672             'params' => ['do' => $type],
673             'nofollow' => true,
674             'replacement' => ''
675         ];
676         $unknown = true;
677     }
678 
679     $evt = new Event('TPL_ACTION_GET', $data);
680     if ($evt->advise_before()) {
681         //handle unknown types
682         if ($unknown) {
683             $data = '[unknown %s type]';
```

```
684         }
685     }
686     $evt->advise_after();
687     unset($evt);
688 
689     return $data;
690 }
691 
692/**
693* Обертка вокруг tpl_button() и tpl_actionlink()
694*
695* @param  string $ тип действие команда
696* @param  bool $ ссылка ссылка или кнопка формы?
697* @param  string | bool $ wrapper Обертка HTML-элемента
698* @param  bool $ return return или print
699* @param  string $ pre префикс для ссылок
700* @param  string $ suf суффикс для ссылок
701* @param  string $ внутренний внутренний HTML для ссылок
702* @return  bool | строка
703*
704* @автор Аника Хенке <anika@selfthinker.org>
705* @deprecated 2017-09-01 см. devel:menus
706*/
707 function tpl_action($type, $link = false, $wrapper = false, $return =
false, $pre = '', $suf = '', $inner = '')
708 {
709     dbg_deprecated('see devel:menus');
710     $out = '';
711     if ($link) {
712         $out .= tpl_actionlink($type, $pre, $suf, $inner, true);
713     } else {
714         $out .= tpl_button($type, true);
715     }
716     if ($out && $wrapper) $out = "<$wrapper>$out</$wrapper>";
717 
718     if ($return) return $out;
719     echo $out;
720     return (bool)$out;
721 }
722 
723/**
724* Распечатать форму поиска
725*
726* Если первый параметр задан как div с идентификатором 'qsearch_out', то будет
727* добавляется, который инструктирует страницу ajax quicksearch включиться и
разместить
728* его вывод в этот div. Второй параметр управляет собственным
729* атрибут автозаполнения. Если установлено значение false, этот атрибут будет
установлен с
730* значение "off" указывает браузеру отключить встроенные функции
731* функция автодополнения (MSIE и Firefox)
```

```
732*
733* @param  bool $ ajax
734* @param  bool $ автозаполнение
735* @return  bool
736*
737* @автор Андреас Гор <andi@splitbrain.org>
738*/
739 function tpl_searchform($ajax = true, $autocomplete = true)
740 {
741     global $lang;
742     global $ACT;
743     global $QUERY;
744     global $ID;
745
746     // don't print the search form if search action has been disabled
747     if (!actionOK('search')) return false;
748
749     $searchForm = new Form([
750         'action' => wl(),
751         'method' => 'get',
752         'role' => 'search',
753         'class' => 'search',
754         'id' => 'dw__search',
755     ], true);
756     $searchForm->addTagOpen('div')->addClass('no');
757     $searchForm->setHiddenField('do', 'search');
758     $searchForm->setHiddenField('id', $ID);
759     $searchForm->addTextInput('q')
760         ->addClass('edit')
761         ->attrs([
762             'title' => '[F]',
763             'accesskey' => 'f',
764             'placeholder' => $lang['btn_search'],
765             'autocomplete' => $autocomplete ? 'on' : 'off',
766         ])
767         ->id('qsearch__in')
768         ->val($ACT === 'search' ? $QUERY : '')
769         ->useInput(false);
770     $searchForm->addButton('', $lang['btn_search'])->attrs([
771         'type' => 'submit',
772         'title' => $lang['btn_search'],
773     ]);
774     if ($ajax) {
775
$searchForm->addTagOpen('div')->id('qsearch__out')->addClass('ajax_qsearch
JSpopup');
776         $searchForm->addTagClose('div');
777     }
778     $searchForm->addTagClose('div');
779
780     echo $searchForm->toHTML('QuickSearch');
```

```php

    return true;
}

/**
* Распечатать след навигационной цепочки
*
* @param  string $ sep Разделитель между записями
* @param  bool $ return return или print
* @return  bool | строка
*
* @автор Андреас Гор <andi@splitbrain.org>
*/
function tpl_breadcrumbs($sep = null, $return = false)
{
    global $lang;
    global $conf;

    //check if enabled
    if (!$conf['breadcrumbs']) return false;

    //set default
    if (is_null($sep)) $sep = '•';

    $out = '';

    $crumbs = breadcrumbs(); //setup crumb trace

    $crumbs_sep = ' <span class="bcsep">' . $sep . '</span> ';

    //render crumbs, highlight the last one
    $out .= '<span class="bchead">' . $lang['breadcrumb'] . '</span>';
    $last = count($crumbs);
    $i = 0;
    foreach ($crumbs as $id => $name) {
        $i++;
        $out .= $crumbs_sep;
        if ($i == $last) $out .= '<span class="curid">';
        $out .= '<bdi>' . tpl_link(wl($id), hsc($name),
'class="breadcrumbs" title="' . $id . '"', true) . '</bdi>';
        if ($i == $last) $out .= '</span>';
    }
    if ($return) return $out;
    echo $out;
    return (bool)$out;
}

/**
* Иерархическая навигационная цепочка
*
* Этот код был предложен в качестве замены обычным хлебным крошкам.
```

```
831* Имеет смысл только при наличии глубокой структуры сайта.
832*
833* @param  string $ sep Разделитель между записями
834* @param  bool $ return return или print
835* @return  bool | строка
836*
837* @todo     может вести себя странно в языках с письмом справа налево
838* @автор <fredrik@averpil.com>
839* @автор Андреас Гор <andi@splitbrain.org>
840* @автор Найджел Макни <oracle.shinoda@gmail.com>
841* @автор Шон Коутс <sean@caedmon.net>
842*/
843 function tpl_youarehere($sep = null, $return = false)
844 {
845     global $conf;
846     global $ID;
847     global $lang;
848
849     // check if enabled
850     if (!$conf['youarehere']) return false;
851
852     //set default
853     if (is_null($sep)) $sep = ' » ';
854
855     $out = '';
856
857     $parts = explode(':', $ID);
858     $count = count($parts);
859
860     $out .= '<span class="bchead">' . $lang['youarehere'] . ' </span>';
861
862     // always print the startpage
863     $out .= '<span class="home">' . tpl_pagelink(':' . $conf['start'],
null, true) . '</span>';
864
865     // print intermediate namespace links
866     $part = '';
867     for ($i = 0; $i < $count - 1; $i++) {
868         $part .= $parts[$i] . ':';
869         $page = $part;
870         if ($page == $conf['start']) continue; // Skip startpage
871
872         // output
873         $out .= $sep . tpl_pagelink($page, null, true);
874     }
875
876     // print current page, skipping start page, skipping for namespace
index
877     if (isset($page)) {
878         $page = (new PageResolver('root'))->resolveId($page);
879         if ($page == $part . $parts[$i]) {
```

```
880             if ($return) return $out;
881             echo $out;
882             return true;
883         }
884     }
885     $page = $part . $parts[$i];
886     if ($page == $conf['start']) {
887         if ($return) return $out;
888         echo $out;
889         return true;
890     }
891     $out .= $sep;
892     $out .= tpl_pagelink($page, null, true);
893     if ($return) return $out;
894     echo $out;
895     return (bool)$out;
896 }
897
898/**
899* Распечатать информацию, если пользователь вошел в систему
900* и в этом случае показывать полное имя
901*
902* Можно ли в будущем добавить ссылку на профиль?
903*
904* @return  bool
905*
906* @автор Андреас Гор <andi@splitbrain.org>
907*/
908 function tpl_userinfo()
909 {
910     global $lang;
911     /** @var Input $INPUT */
912     global $INPUT;
913
914     if ($INPUT->server->str('REMOTE_USER')) {
915         echo $lang['loggedinas'] . ' ' . userlink();
916         return true;
917     }
918     return false;
919 }
920
921/**
922* Распечатать некоторую информацию о текущей странице
923*
924* @param  bool $ ret возвращает содержимое вместо его печати
925* @return  bool | строка
926*
927* @автор Андреас Гор <andi@splitbrain.org>
928*/
929 function tpl_pageinfo($ret = false)
930 {
```

```
931     global $conf;
932     global $lang;
933     global $INFO;
934     global $ID;
935 
936     // return if we are not allowed to view the page
937     if (!auth_quickaclcheck($ID)) {
938         return false;
939     }
940 
941     // prepare date and path
942     $fn = $INFO['filepath'];
943     if (!$conf['fullpath']) {
944         if ($INFO['rev']) {
945             $fn = str_replace($conf['olddir'] . '/', '', $fn);
946         } else {
947             $fn = str_replace($conf['datadir'] . '/', '', $fn);
948         }
949     }
950     $fn = utf8_decodeFN($fn);
951     $date = dformat($INFO['lastmod']);
952 
953     // print it
954     if ($INFO['exists']) {
955         $out = '<bdi>' . $fn . '</bdi>';
956         $out .= ' · ';
957         $out .= $lang['lastmod'];
958         $out .= ' ';
959         $out .= $date;
960         if ($INFO['editor']) {
961             $out .= ' ' . $lang['by'] . ' ';
962             $out .= '<bdi>' . editorinfo($INFO['editor']) . '</bdi>';
963         } else {
964             $out .= ' (' . $lang['external_edit'] . ')';
965         }
966         if ($INFO['locked']) {
967             $out .= ' · ';
968             $out .= $lang['lockedby'];
969             $out .= ' ';
970             $out .= '<bdi>' . editorinfo($INFO['locked']) . '</bdi>';
971         }
972         if ($ret) {
973             return $out;
974         } else {
975             echo $out;
976             return true;
977         }
978     }
979     return false;
980 }
981 
```

```php
/**
* Печатает или возвращает имя указанной страницы (текущей, если не указано).
*
* Если включено использование заголовка, будет использоваться первый заголовок, в противном случае
* используется указанный идентификатор.
*
* @param  string $ id идентификатор страницы
* @param  bool $ ret возвращает содержимое вместо печати
* @return  bool | строка
*
* @автор Андреас Гор <andi@splitbrain.org>
*/
function tpl_pagetitle($id = null, $ret = false)
{
    global $ACT, $conf, $lang;

    if (is_null($id)) {
        global $ID;
        $id = $ID;
    }

    $name = $id;
    if (useHeading('navigation')) {
        $first_heading = p_get_first_heading($id);
        if ($first_heading) $name = $first_heading;
    }

    // default page title is the page name, modify with the current action
    switch ($ACT) {
        // admin functions
        case 'admin':
            $page_title = $lang['btn_admin'];
            // try to get the plugin name
            /** @var AdminPlugin $plugin */
            if ($plugin = plugin_getRequestAdminPlugin()) {
                $plugin_title = $plugin->getMenuText($conf['lang']);
                $page_title = $plugin_title ?: $plugin->getPluginName();
            }
            break;

        // show action as title
        case 'login':
        case 'profile':
        case 'register':
        case 'resendpwd':
        case 'index':
        case 'search':
            $page_title = $lang['btn_' . $ACT];
```

```
1030             break;
1031
1032         // add pen during editing
1033         case 'edit':
1034         case 'preview':
1035             $page_title = "✎ " . $name;
1036             break;
1037
1038         // add action to page name
1039         case 'revisions':
1040             $page_title = $name . ' - ' . $lang['btn_revs'];
1041             break;
1042
1043         // add action to page name
1044         case 'backlink':
1045         case 'recent':
1046         case 'subscribe':
1047             $page_title = $name . ' - ' . $lang['btn_' . $ACT];
1048             break;
1049
1050         default: // SHOW and anything else not included
1051             $page_title = $name;
1052     }
1053
1054     if ($ret) {
1055         return hsc($page_title);
1056     } else {
1057         echo hsc($page_title);
1058         return true;
1059     }
1060 }
1061
1062/**
1063* Возвращает запрошенный тег EXIF / IPTC из текущего изображения
1064*
1065* Если $tags — это массив, то все заданные теги проверяются до тех пор, пока не
будет найден
1066* значение найдено. Если значение не найдено, возвращается $alt.
1067*
1068* Какие тексты известны, определяется в функциях _exifTagNames
1069* и _iptcTagNames() в inc / jpeg.php (Вам необходимо добавить IPTC
1070* к именам последнего)
1071*
1072* Разрешено только в: detail.php
1073*
1074* @param  array | string $ tags тег или массив тегов для проверки
1075* @param  string $ alt альтернативный вывод, если данные не найдены
1076* @param  null | string $ src источник изображения, если не указан,
используется глобальный $SRC
1077* @возвращаемая строка
1078*
```

```
1079* @автор Андреас Гор <andi@splitbrain.org>
1080*/
1081 function tpl_img_getTag($tags, $alt = '', $src = null)
1082 {
1083     // Init Exif Reader
1084     global $SRC, $imgMeta;
1085
1086     if (is_null($src)) $src = $SRC;
1087     if (is_null($src)) return $alt;
1088
1089     if (!isset($imgMeta)) {
1090         $imgMeta = new JpegMeta($src);
1091     }
1092     if ($imgMeta === false) return $alt;
1093     $info = cleanText($imgMeta->getField($tags));
1094     if (!$info) return $alt;
1095     return $info;
1096 }
1097
1098
1099/**
1100* Мусор собирает открытый объект JpegMeta.
1101*/
1102 function tpl_img_close()
1103 {
1104     global $imgMeta;
1105     $imgMeta = null;
1106 }
1107
1108/**
1109* Выводит HTML-список описаний метатегов текущего изображения
1110*/
1111 function tpl_img_meta()
1112 {
1113     global $lang;
1114
1115     $tags = tpl_get_img_meta();
1116
1117     echo '<dl>';
1118     foreach ($tags as $tag) {
1119         $label = $lang[$tag['langkey']];
1120         if (!$label) $label = $tag['langkey'] . ':';
1121
1122         echo '<dt>' . $label . '</dt><dd>';
1123         if ($tag['type'] == 'date') {
1124             echo dformat($tag['value']);
1125         } else {
1126             echo hsc($tag['value']);
1127         }
1128         echo '</dd>';
1129     }
```

```
1130     echo '</dl>';
1131 }
1132
1133/**
1134* Возвращает метаданные, настроенные в файле конфигурации mediameta, готовые для создания html
1135*
1136* @return   массив с массивами, содержащими записи:
1137* - строка langkey key для поиска в переменной $lang, если не найдена, выводится как есть
1138* - строковый тип значения
1139* - строковое значение тега (неэкранированное)
1140*/
1141 function tpl_get_img_meta()
1142 {
1143
1144     $config_files = getConfigFiles('mediameta');
1145     foreach ($config_files as $config_file) {
1146         if (file_exists($config_file)) {
1147             include($config_file);
1148         }
1149     }
1150     $tags = [];
1151     foreach ($fields as $tag) {
1152         $t = [];
1153         if (!empty($tag[0])) {
1154             $t = [$tag[0]];
1155         }
1156         if (isset($tag[3]) && is_array($tag[3])) {
1157             $t = array_merge($t, $tag[3]);
1158         }
1159         $value = tpl_img_getTag($t);
1160         if ($value) {
1161             $tags[] = ['langkey' => $tag[1], 'type' => $tag[2], 'value'
=> $value];
1162         }
1163     }
1164     return $tags;
1165 }
1166
1167/**
1168* Печатает изображение со ссылкой на полноразмерную версию
1169*
1170* Разрешено только в: detail.php
1171*
1172* @triggers TPL_IMG_DISPLAY
1173* @param  int $ maxwidth - максимальная ширина изображения
1174* @param  int $ maxheight - максимальная высота изображения
1175* @param  bool $ link - ссылка на исходный размер?
1176* @param  array $ params - дополнительные атрибуты изображения
1177* @return  bool Результат TPL_IMG_DISPLAY
```

```php
1178*/
1179 function tpl_img($maxwidth = 0, $maxheight = 0, $link = true, $params =
null)
1180 {
1181     global $IMG;
1182     /** @var Input $INPUT */
1183     global $INPUT;
1184     global $REV;
1185     $w = (int)tpl_img_getTag('File.Width');
1186     $h = (int)tpl_img_getTag('File.Height');
1187
1188     //изменить размер до указанных максимальных значений
1189     $ratio = 1;
1190     if ($w >= $h) {
1191         if ($maxwidth && $w >= $maxwidth) {
1192             $ratio = $maxwidth / $w;
1193         } elseif ($maxheight && $h > $maxheight) {
1194             $ratio = $maxheight / $h;
1195         }
1196     } elseif ($maxheight && $h >= $maxheight) {
1197         $ratio = $maxheight / $h;
1198     } elseif ($maxwidth && $w > $maxwidth) {
1199         $ratio = $maxwidth / $w;
1200     }
1201     if ($ratio) {
1202         $w = floor($ratio * $w);
1203         $h = floor($ratio * $h);
1204     }
1205
1206     //подготовить URL-адреса
1207     $url = ml($IMG, ['cache' => $INPUT->str('cache'), 'rev' => $REV],
true, '&');
1208     $src = ml($IMG, ['cache' => $INPUT->str('cache'), 'rev' => $REV,
'w' => $w, 'h' => $h], true, '&');
1209
1210     //подготовить атрибуты
1211     $alt = tpl_img_getTag('Simple.Title');
1212     if (is_null($params)) {
1213         $p = [];
1214     } else {
1215         $p = $params;
1216     }
1217     if ($w) $p['width'] = $w;
1218     if ($h) $p['height'] = $h;
1219     $p['class'] = 'img_detail';
1220     if ($alt) {
1221         $p['alt'] = $alt;
1222         $p['title'] = $alt;
1223     } else {
1224         $p['alt'] = '';
1225     }
```

```
1226     $p['src'] = $src;
1227
1228     $data = ['url' => ($link ? $url : null), 'params' => $p];
1229     return Event::createAndTrigger('TPL_IMG_DISPLAY', $data,
'_tpl_img_action', true);
1230 }
1231
1232/**
1233* Действие по умолчанию для  TPL_IMG_DISPLAY
1234*
1235* @param  массив $ данные
1236* @return  bool
1237*/
1238 function _tpl_img_action($data)
1239 {
1240     global $lang;
1241     $p = buildAttributes($data['params']);
1242
1243     if ($data['url']) echo '<a href="' . hsc($data['url']) . '"
title="' . $lang['mediaview'] . '">';
1244     echo '<img ' . $p . '/>';
1245     if ($data['url']) echo '</a>';
1246     return true;
1247 }
1248
1249/**
1250* Эта функция вставляет небольшой  gif-файл, который на самом деле является
функцией индексатора.
1251*
1252* Должен вызываться где-то в самом конце шаблона  main.php
1253*
1254* @return  bool
1255*/
1256 function tpl_indexerWebBug()
1257 {
1258     global $ID;
1259
1260     $p = [];
1261     $p['src'] = DOKU_BASE . 'lib/exe/taskrunner.php?id=' .
rawurlencode($ID) .
1262         '&' . time();
1263     $p['width'] = 2; //no more 1x1 px image because we live in times of
ad blockers...
1264     $p['height'] = 1;
1265     $p['alt'] = '';
1266     $att = buildAttributes($p);
1267     echo "<img $att />";
1268     return true;
1269 }
1270
1271/**
```

```php
/* tpl_getConf($id)
*
* используйте эту функцию для доступа к переменным конфигурации шаблона
*
* @param  string $ id имя значения для доступа
* @param  mixed $ notset что возвращать, если настройка недоступна
* @return  смешанный
*/
function tpl_getConf($id, $notset = false)
{
    global $conf;
    static $tpl_configloaded = false;

    $tpl = $conf['template'];

    if (!$tpl_configloaded) {
        $tconf = tpl_loadConfig();
        if ($tconf !== false) {
            foreach ($tconf as $key => $value) {
                if (isset($conf['tpl'][$tpl][$key])) continue;
                $conf['tpl'][$tpl][$key] = $value;
            }
            $tpl_configloaded = true;
        }
    }

    return $conf['tpl'][$tpl][$id] ?? $notset;
}

/**
* tpl_loadConfig()
*
* считывает все переменные конфигурации шаблона
* эта функция автоматически вызывается  tpl_getConf()
*
* @return  false | массив
*/
function tpl_loadConfig()
{

    $file = tpl_incdir() . '/conf/default.php';
    $conf = [];

    if (!file_exists($file)) return false;

   // загрузить файл конфигурации по умолчанию
    include($file);

    return $conf;
}

```

```
1323// методы языка
1324
1323// методы языка
1324
1325/**
1326* tpl_getLang($id)
1327*
1328* используйте эту функцию для доступа к переменным языка шаблона
1329*
1330* @param  string $ id ключ языковой строки
1331* @возвращаемая строка
1332*/
1333 function tpl_getLang($id)
1334 {
1335     static $lang = [];
1336
1337     if (count($lang) === 0) {
1338         global $conf, $config_cascade; // definitely don't invoke
"global $lang"
1339
1340         $path = tpl_incdir() . 'lang/';
1341
1342         $lang = [];
1343
1344        // не включайте один раз
1345         @include($path . 'en/lang.php');
1346         foreach ($config_cascade['lang']['template'] as $config_file) {
1347             if (file_exists($config_file . $conf['template'] .
'/en/lang.php')) {
1348                 include($config_file . $conf['template'] .
'/en/lang.php');
1349             }
1350         }
1351
1352         if ($conf['lang'] != 'en') {
1353             @include($path . $conf['lang'] . '/lang.php');
1354             foreach ($config_cascade['lang']['template'] as
$config_file) {
1355                 if (file_exists($config_file . $conf['template'] . '/'
. $conf['lang'] . '/lang.php')) {
1356                     include($config_file . $conf['template'] . '/' .
$conf['lang'] . '/lang.php');
1357                 }
1358             }
1359         }
1360     }
1361     return $lang[$id] ?? '';
1362 }
1363
1364/**
1365* Извлечь файл, зависящий от языка, и передать его в  xhtml-рендерер для
```

```
отображения
1366* эквивалент шаблона p_locale_xhtml()
1367*
1368* @param  string $ id идентификатор страницы вики, зависящей от языка
1369* @return   string      анализирует содержимое страницы вики в формате xhtml
1370*/
1371 function tpl_locale_xhtml($id)
1372 {
1373     return p_cached_output(tpl_localeFN($id));
1374 }
1375
1376/**
1377* Добавляет соответствующий путь к имени файла, зависящему от языка
1378*
1379* @param  string $ id идентификатор локализованного текста
1380* @return  string вики-текст
1381*/
1382 function tpl_localeFN($id)
1383 {
1384     $path = tpl_incdir() . 'lang/';
1385     global $conf;
1386     $file = DOKU_CONF . 'template_lang/' . $conf['template'] . '/' .
$conf['lang'] . '/' . $id . '.txt';
1387     if (!file_exists($file)) {
1388         $file = $path . $conf['lang'] . '/' . $id . '.txt';
1389         if (!file_exists($file)) {
1390             //fall back to english
1391             $file = $path . 'en/' . $id . '.txt';
1392         }
1393     }
1394     return $file;
1395 }
1396
1397/**
1398* выводит «основной контент» во всплывающем окне медиаменеджера
1399*
1400* В зависимости от действий пользователя это может быть список
1401* файлы в пространстве имен, диалоговое окно редактирования метаданных или
1402* сообщение о ссылках на страницы
1403*
1404* Разрешено только в mediamanager.php
1405*
1406* @triggers МЕДИАМЕНЕДЖЕР_КОНТЕНТ_ВЫВОД
1407* @param  bool $ fromajax - установить true при вызове этой функции через
ajax
1408* @param  string $ сортировка
1409*
1410* @автор Андреас Гор <andi@splitbrain.org>
1411*/
1412 function tpl_mediaContent($fromajax = false, $sort = 'natural')
1413 {
```

```
1414     global $IMG;
1415     global $AUTH;
1416     global $INUSE;
1417     global $NS;
1418     global $JUMPTO;
1419     /** @var Input $INPUT */
1420     global $INPUT;
1421 
1422     $do = $INPUT->extract('do')->str('do');
1423     if (in_array($do, ['save', 'cancel'])) $do = '';
1424 
1425     if (!$do) {
1426         if ($INPUT->bool('edit')) {
1427             $do = 'metaform';
1428         } elseif (is_array($INUSE)) {
1429             $do = 'filesinuse';
1430         } else {
1431             $do = 'filelist';
1432         }
1433     }
1434 
1435    // выводим панель содержимого, обернутую в событие.
1436     if (!$fromajax) echo '<div id="media__content">';
1437     $data = ['do' => $do];
1438     $evt = new Event('MEDIAMANAGER_CONTENT_OUTPUT', $data);
1439     if ($evt->advise_before()) {
1440         $do = $data['do'];
1441         if ($do == 'filesinuse') {
1442             media_filesinuse($INUSE, $IMG);
1443         } elseif ($do == 'filelist') {
1444             media_filelist($NS, $AUTH, $JUMPTO, false, $sort);
1445         } elseif ($do == 'searchlist') {
1446             media_searchlist($INPUT->str('q'), $NS, $AUTH);
1447         } else {
1448             msg('Unknown action ' . hsc($do), -1);
1449         }
1450     }
1451     $evt->advise_after();
1452     unset($evt);
1453     if (!$fromajax) echo '</div>';
1454 }
1455 
1456/**
1457* Печатает центральный столбец в полноэкранном медиа-менеджере
1458* В зависимости от открытой вкладки это может быть список
1459* файлы в пространстве имен, форма загрузки или форма поиска
1460*
1461* @author Катя Арзамасцева <pshns@ukr.net>
1462*/
1463 function tpl_mediaFileList()
1464 {
```

```
1465     global $AUTH;
1466     global $NS;
1467     global $JUMPTO;
1468     global $lang;
1469     /** @var Input $INPUT */
1470     global $INPUT;
1471
1472     $opened_tab = $INPUT->str('tab_files');
1473     if (!$opened_tab || !in_array($opened_tab, ['files', 'upload',
'search'])) $opened_tab = 'files';
1474     if ($INPUT->str('mediado') == 'update') $opened_tab = 'upload';
1475
1476     echo '<h2 class="a11y">' . $lang['mediaselect'] . '</h2>' . NL;
1477
1478     media_tabs_files($opened_tab);
1479
1480     echo '<div class="panelHeader">' . NL;
1481     echo '<h3>';
1482     $tabTitle = $NS ?: '[' . $lang['mediaroot'] . ']';
1483     printf($lang['media_' . $opened_tab], '<strong>' . hsc($tabTitle) .
'</strong>');
1484     echo '</h3>' . NL;
1485     if ($opened_tab === 'search' || $opened_tab === 'files') {
1486         media_tab_files_options();
1487     }
1488     echo '</div>' . NL;
1489
1490     echo '<div class="panelContent">' . NL;
1491     if ($opened_tab == 'files') {
1492         media_tab_files($NS, $AUTH, $JUMPTO);
1493     } elseif ($opened_tab == 'upload') {
1494         media_tab_upload($NS, $AUTH, $JUMPTO);
1495     } elseif ($opened_tab == 'search') {
1496         media_tab_search($NS, $AUTH);
1497     }
1498     echo '</div>' . NL;
1499 }
1500
1501/**
1502* Печатает третий столбец в полноэкранном медиа-менеджере
1503* В зависимости от открытой вкладки это могут быть сведения о
1504* выбранный файл, диалоговое окно редактирования метаданных или
1505* список ревизий файлов
1506*
1507* @param  string $ изображение
1508* @param  boolean $ rev
1509*
1510* @author Катя Арзамасцева <pshns@ukr.net>
1511*/
1512 function tpl_mediaFileDetails($image, $rev)
1513 {
```

```
1514     global $conf, $DEL, $lang;
1515     /** @var Input $INPUT */
1516     global $INPUT;
1517
1518     $removed = (
1519         !file_exists(mediaFN($image)) &&
1520         file_exists(mediaMetaFN($image, '.changes')) &&
1521         $conf['mediarevisions']
1522     );
1523     if (!$image || (!file_exists(mediaFN($image)) && !$removed) ||
$DEL) return;
1524     if ($rev && !file_exists(mediaFN($image, $rev))) $rev = false;
1525     $ns = getNS($image);
1526     $do = $INPUT->str('mediado');
1527
1528     $opened_tab = $INPUT->str('tab_details');
1529
1530     $tab_array = ['view'];
1531     [, $mime] = mimetype($image);
1532     if ($mime == 'image/jpeg') {
1533         $tab_array[] = 'edit';
1534     }
1535     if ($conf['mediarevisions']) {
1536         $tab_array[] = 'history';
1537     }
1538
1539     if (!$opened_tab || !in_array($opened_tab, $tab_array)) $opened_tab
= 'view';
1540     if ($INPUT->bool('edit')) $opened_tab = 'edit';
1541     if ($do == 'restore') $opened_tab = 'view';
1542
1543     media_tabs_details($image, $opened_tab);
1544
1545     echo '<div class="panelHeader"><h3>';
1546     [$ext] = mimetype($image, false);
1547     $class = preg_replace('/[^_\-a-z0-9]+/i', '_', $ext);
1548     $class = 'select mediafile mf_' . $class;
1549
1550     $attributes = $rev ? ['rev' => $rev] : [];
1551     $tabTitle = sprintf(
1552         '<strong><a href="%s" class="%s" title="%s">%s</a></strong>',
1553         ml($image, $attributes),
1554         $class,
1555         $lang['mediaview'],
1556         $image
1557     );
1558     if ($opened_tab === 'view' && $rev) {
1559         printf($lang['media_viewold'], $tabTitle, dformat($rev));
1560     } else {
1561         printf($lang['media_' . $opened_tab], $tabTitle);
1562     }
```

```
1563
1564     echo '</h3></div>' . NL;
1565
1566     echo '<div class="panelContent">' . NL;
1567
1568     if ($opened_tab == 'view') {
1569         media_tab_view($image, $ns, null, $rev);
1570     } elseif ($opened_tab == 'edit' && !$removed) {
1571         media_tab_edit($image, $ns);
1572     } elseif ($opened_tab == 'history' && $conf['mediarevisions']) {
1573         media_tab_history($image, $ns);
1574     }
1575
1576     echo '</div>' . NL;
1577 }
1578
1579/**
1580* выводит дерево пространства имен во всплывающем окне медиаменеджера
1581*
1582* Разрешено только в mediamanager.php
1583*
1584* @автор Андреас Гор <andi@splitbrain.org>
1585*/
1586 function tpl_mediaTree()
1587 {
1588     global $NS;
1589     echo '<div id="media__tree">';
1590     media_nstree($NS);
1591     echo '</div>';
1592 }
1593
1594/**
1595* Распечатать выпадающее меню со всеми действиями DokuWiki
1596*
1597* Примечание: здесь не будут использоваться красивые URL-адреса.
1598*
1599* @param  string $ пусто пустая метка параметра
1600* @param  string $ кнопка подписи кнопки отправки
1601*
1602* @автор Андреас Гор <andi@splitbrain.org>
1603* @deprecated 2017-09-01 см. devel:menus
1604*/
1605 function tpl_actiondropdown($empty = '', $button = '&gt;')
1606 {
1607     dbg_deprecated('see devel:menus');
1608     $menu = new MobileMenu();
1609     echo $menu->getDropdown($empty, $button);
1610 }
1611
1612/**
1613* Распечатать информационную строку об использованной лицензии
```

```
1614*
1615* @param  string $ img распечатать изображение? (|кнопка|значок)
1616* @param  bool $ imgonly пропустить текстовое описание?
1617* @param  bool $ return, если true, не печатать, а возвращать HTML
1618* @param  bool $ обернуть в div с class="license"?
1619* @возвращаемая строка
1620*
1621* @автор Андреас Гор <andi@splitbrain.org>
1622*/
1623 function tpl_license($img = 'badge', $imgonly = false, $return = false,
$wrap = true)
1624 {
1625     global $license;
1626     global $conf;
1627     global $lang;
1628     if (!$conf['license']) return '';
1629     if (!is_array($license[$conf['license']])) return '';
1630     $lic = $license[$conf['license']];
1631     $target = ($conf['target']['extern']) ? ' target="' .
$conf['target']['extern'] . '"' : '';
1632
1633     $out = '';
1634     if ($wrap) $out .= '<div class="license">';
1635     if ($img) {
1636         $src = license_img($img);
1637         if ($src) {
1638             $out .= '<a href="' . $lic['url'] . '" rel="license"' .
$target;
1639             $out .= '><img src="' . DOKU_BASE . $src . '" alt="' .
$lic['name'] . '" /></a>';
1640             if (!$imgonly) $out .= ' ';
1641         }
1642     }
1643     if (!$imgonly) {
1644         $out .= $lang['license'] . ' ';
1645         $out .= '<bdi><a href="' . $lic['url'] . '" rel="license"
class="urlextern"' . $target;
1646         $out .= '>' . $lic['name'] . '</a></bdi>';
1647     }
1648     if ($wrap) $out .= '</div>';
1649
1650     if ($return) return $out;
1651     echo $out;
1652     return '';
1653 }
1654
1655/**
1656* Включает визуализированный HTML-код указанной страницы
1657*
1658* Эта функция полезна для заполнения боковых панелей или подобных функций в
1659* шаблон
```

```
1660*
1661* @param  string $ pageid Имя страницы, которую вы хотите включить
1662* @param  bool $ print Следует ли печатать содержимое или только возвращать
его
1663* @param  bool $ propagate Искать также и в более высоких пространствах имен?
1664* @param  bool $ useacl Включать страницу только в том случае, если списки
контроля доступа проверены?
1665* @return  bool | null | строка
1666*/
1667 function tpl_include_page($pageid, $print = true, $propagate = false,
$useacl = true)
1668 {
1669     if ($propagate) {
1670         $pageid = page_findnearest($pageid, $useacl);
1671     } elseif ($useacl && auth_quickaclcheck($pageid) == AUTH_NONE) {
1672         return false;
1673     }
1674     if (!$pageid) return false;
1675
1676     global $TOC;
1677     $oldtoc = $TOC;
1678     $html = p_wiki_xhtml($pageid, '', false);
1679     $TOC = $oldtoc;
1680
1681     if ($print) echo $html;
1682     return $html;
1683 }
1684
1685/**
1686* Отобразить форму подписки
1687*
1688* @автор Адриан Лэнг <lang@cosmocode.de>
1689* @устаревший 2020-07-23
1690*/
1691 function tpl_subscribe()
1692 {
1693     dbg_deprecated(Subscribe::class . '::show()');
1694     (new Subscribe())->show();
1695 }
1696
1697/**
1698* Пытается отправить уже созданный контент прямо в браузер
1699*
1700* Оборачивает ob_flush() и flush()
1701*
1702* @автор Андреас Гор <andi@splitbrain.org>
1703*/
1704 function tpl_flush()
1705 {
1706     if (ob_get_level() > 0) ob_flush();
1707     flush();
```

```
1708 }
1709
1710/**
1711* Пытается найти файл ресурсов в указанных местах.
1712*
1713* Если указанное местоположение начинается с двоеточия, предполагается, что это
медиа
1714* файл, в противном случае предполагается, что он относится к текущему шаблону
1715*
1716* @параметр  строка []$ поиск мест для просмотра
1717* @param  bool $ abs , если использовать абсолютный URL
1718* @param  массив     &$imginfo заполнен с помощью getimagesize()
1719* @param  bool $ fallback использовать резервное изображение, если цель не
найдена, или вернуть «false», если она потенциальная
1720* требуется ложный результат
1721* @возвращаемая строка
1722*
1723* @автор Андреас Гор <andi@splitbrain.org>
1724*/
1725 function tpl_getMediaFile($search, $abs = false, &$imginfo = null,
$fallback = true)
1726 {
1727     $img = '';
1728     $file = '';
1729     $ismedia = false;
1730     // перебираем кандидатов, пока не будет найдено совпадение:
1731     foreach ($search as $img) {
1732         if (str_starts_with($img, ':')) {
1733             $file = mediaFN($img);
1734             $ismedia = true;
1735         } else {
1736             $file = tpl_incdir() . $img;
1737             $ismedia = false;
1738         }
1739
1740         if (file_exists($file)) break;
1741     }
1742
1743     // управлять несуществующей целью
1744     if (!file_exists($file)) {
1745         // дать результат для резервного изображения
1746         if ($fallback) {
1747             $file = DOKU_INC . 'lib/images/blank.gif';
1748             // остановить процесс, если требуется ложный результат (если
$fallback равен false)
1749         } else {
1750             return false;
1751         }
1752     }
1753
1754     // извлечь данные изображения, если требуется
```

```
1755     if (!is_null($imginfo)) {
1756         $imginfo = getimagesize($file);
1757     }
1758
1759     // создать URL
1760     if ($ismedia) {
1761         $url = ml($img, '', true, '', $abs);
1762     } else {
1763         $url = tpl_basedir() . $img;
1764         if ($abs) $url = DOKU_URL . substr($url, strlen(DOKU_REL));
1765     }
1766
1767     return $url;
1768 }
1769
1770/**
1771* PHP включает файл
1772*
1773* либо из каталога conf, если он существует, в противном случае используйте
1774* файл в корневом каталоге шаблона.
1775*
1776* Функция учитывает настройки каскада конфигураций и ищет заданные
1777* файл рядом с «основными» файлами конфигурации, в порядке защищенный,
локальный,
1778* по умолчанию.
1779*
1780* Примечание: здесь не выполняется экранирование или проверка на работоспособность.
Никогда не передавайте пользовательский ввод
1781* к этой функции!
1782*
1783* @param string $ файл
1784*
1785* @автор Андреас Гор <andi@splitbrain.org>
1786* @автор Аника Хенке <anika@selfthinker.org>
1787*/
1788 function tpl_includeFile($file)
1789 {
1790     global $config_cascade;
1791     foreach (['protected', 'local', 'default'] as $config_group) {
1792         if (empty($config_cascade['main'][$config_group])) continue;
1793         foreach ($config_cascade['main'][$config_group] as $conf_file)
{
1794             $dir = dirname($conf_file);
1795             if (file_exists("$dir/$file")) {
1796                 include("$dir/$file");
1797                 return;
1798             }
1799         }
1800     }
1801
1802     // все еще здесь? попробуйте шаблон dir
```

```
1803     $file = tpl_incdir() . $file;
1804     if (file_exists($file)) {
1805         include($file);
1806     }
1807 }
1808
1809/**
1810* Возвращает тег <link> для различных типов иконок (favicon|mobile|generic)
1811*
1812* @param  array $ types - список типов иконок для отображения
(favicon|mobile|generic)
1813* @возвращаемая строка
1814*
1815* @автор Аника Хенке <anika@selfthinker.org>
1816*/
1817 function tpl_favicon($types = ['favicon'])
1818 {
1819
1820     $return = '';
1821
1822     foreach ($types as $type) {
1823         switch ($type) {
1824             case 'favicon':
1825                 $look = [':wiki:favicon.ico', ':favicon.ico',
'images/favicon.ico'];
1826                 $return .= '<link rel="shortcut icon" href="' .
tpl_getMediaFile($look) . '" />' . NL;
1827                 break;
1828             case 'mobile':
1829                 $look = [':wiki:apple-touch-icon.png', ':apple-touch-
icon.png', 'images/apple-touch-icon.png'];
1830                 $return .= '<link rel="apple-touch-icon" href="' .
tpl_getMediaFile($look) . '" />' . NL;
1831                 break;
1832             case 'generic':
1833                 // ideal world solution, which doesn't work in any
browser yet
1834                 $look = [':wiki:favicon.svg', ':favicon.svg',
'images/favicon.svg'];
1835                 $return .= '<link rel="icon" href="' .
tpl_getMediaFile($look) . '" type="image/svg+xml" />' . NL;
1836                 break;
1837         }
1838     }
1839
1840     return $return;
1841 }
1842
1843/**
1844* Печать полноэкранного медиа-менеджера
1845*
```

```
1846* @author Катя Арзамасцева <pshns@ukr.net>
1847*/
1848 function tpl_media()
1849 {
1850     global $NS, $IMG, $JUMPTO, $REV, $lang, $fullscreen, $INPUT;
1851     $fullscreen = true;
1852     require_once DOKU_INC . 'lib/exe/mediamanager.php';
1853
1854     $rev = '';
1855     $image = cleanID($INPUT->str('image'));
1856     if (isset($IMG)) $image = $IMG;
1857     if (isset($JUMPTO)) $image = $JUMPTO;
1858     if (isset($REV) && !$JUMPTO) $rev = $REV;
1859
1860     echo '<div id="mediamanager__page">' . NL;
1861     echo '<h1>' . $lang['btn_media'] . '</h1>' . NL;
1862     html_msgarea();
1863
1864     echo '<div class="panel namespaces">' . NL;
1865     echo '<h2>' . $lang['namespaces'] . '</h2>' . NL;
1866     echo '<div class="panelHeader">';
1867     echo $lang['media_namespaces'];
1868     echo '</div>' . NL;
1869
1870     echo '<div class="panelContent" id="media__tree">' . NL;
1871     media_nstree($NS);
1872     echo '</div>' . NL;
1873     echo '</div>' . NL;
1874
1875     echo '<div class="panel filelist">' . NL;
1876     tpl_mediaFileList();
1877     echo '</div>' . NL;
1878
1879     echo '<div class="panel file">' . NL;
1880     echo '<h2 class="a11y">' . $lang['media_file'] . '</h2>' . NL;
1881     tpl_mediaFileDetails($image, $rev);
1882     echo '</div>' . NL;
1883
1884     echo '</div>' . NL;
1885 }
1886
1887/**
1888* Возвращаем полезные классы макета
1889*
1890* @возвращаемая  строка
1891*
1892* @автор Аника Хенке <anika@selfthinker.org>
1893*/
1894 function tpl_classes()
1895 {
1896     global $ACT, $conf, $ID, $INFO;
```

```
1897     /** @var Input $INPUT */
1898     global $INPUT;
1899 
1900     $classes = [
1901         'dokuwiki',
1902         'mode_' . $ACT,
1903         'tpl_' . $conf['template'],
1904         $INPUT->server->bool('REMOTE_USER') ? 'loggedIn' : '',
1905         (isset($INFO['exists']) && $INFO['exists']) ? '' : 'notFound',
1906         ($ID == $conf['start']) ? 'home' : ''
1907     ];
1908     return implode(' ', $classes);
1909 }
1910 
1911/**
1912* Создать событие для меню инструментов
1913*
1914* @param  string $ toolsname имя меню
1915* @param  массив $ элементы
1916* @param  string $ view например 'main', 'detail', ...
1917*
1918* @автор Аника Хенке <anika@selfthinker.org>
1919* @deprecated 2017-09-01 см. devel:menus
1920*/
1921 function tpl_toolsevent($toolsname, $items, $view = 'main')
1922 {
1923     dbg_deprecated('see devel:menus');
1924     $data = ['view' => $view, 'items' => $items];
1925 
1926     $hook = 'TEMPLATE_' . strtoupper($toolsname) . '_DISPLAY';
1927     $evt = new Event($hook, $data);
1928     if ($evt->advise_before()) {
1929         foreach ($evt->data['items'] as $html) echo $html;
1930     }
1931     $evt->advise_after();
1932 }
1933 
```

Полный перевод для понимая

```
1<?php
2 
3/**
4* Функции шаблонов DokuWiki
5*
6* @license     GPL 2 ( http://www.gnu.org/licenses/gpl.html)
7* @автор   Андреас Гор <andi@splitbrain.org>
8*/
9 
10использовать  dokuwiki \ ActionRouter ;
11используйте  dokuwiki \ Действие \ Исключение \ FatalException ;
```

```
12использовать  dokuwiki \ Extension \ PluginInterface ;
13используйте  dokuwiki \ Ui \ Admin ;
14использовать  dokuwiki \ StyleUtils ;
15использовать  dokuwiki \ Меню\Элемент \ AbstractItem ;
16использовать  dokuwiki \ Форма\Форма ;
17используйте  dokuwiki \ Menu \ MobileMenu ;
18используйте  dokuwiki \ Ui \ Subscribe ;
19используйте  dokuwiki \ Extension \ AdminPlugin ;
20использовать  dokuwiki \ Extension \ Event ;
21используйте  dokuwiki \ File \ PageResolver ;
22
23/**
24* Доступ к файлу шаблона
25*
26* Возвращает путь к указанному файлу внутри текущего шаблона, использует
27* шаблон по умолчанию, если пользовательская версия не существует.
28*
29* @param  string $ файл
30* @возвращаемая строка
31*
32* @автор Андреас Гор <andi@splitbrain.org>
33*/
34 шаблон функции ($ file )
35{
36  глобальная $ conf ;
37
38  если (@ is_readable ( DOKU_INC . ' lib / tpl /' . $ conf [ 'template' ] .
'/' . $ file ))
39    вернуть  DOKU_INC.'lib / tpl /'.$ conf [ ' template' ] . ' /' . $ file
;
40
41  вернуть  DOKU_INC.'lib / tpl / dokuwiki /'.$ file ;
□□□□42}
43
44/**
45* Удобная функция для доступа к каталогу шаблонов из локальной ФС
46*
47* Заменяет устаревшую константу  DOKU_TPLINC.
48*
49* @param  string $ tpl Шаблон для использования, по умолчанию текущий
50* @возвращаемая строка
51*
52* @автор Андреас Гор <andi@splitbrain.org>
53*/
54функция  tpl_incdir ($ tpl = '' )
55{
56  глобальная $ conf ;
57  если (!$ tpl ) $ tpl = $ conf [ 'шаблон' ];
58  вернуть  DOKU_INC . ' lib / tpl /' . $ tpl . '/' ;
59}
60
```

```
61/**
62* Удобная функция доступа к каталогу шаблонов из Интернета
63*
64* Заменяет устаревшую константу DOKU_TPL
65*
66* @param  string $ tpl Шаблон для использования, по умолчанию текущий
67* @возвращаемая строка
68*
69* @автор Андреас Гор <andi@splitbrain.org>
70*/
71функция  tpl_basedir ($ tpl = '' )
72{
73  глобальная $ conf ;
74  если (!$ tpl ) $ tpl = $ conf [ 'шаблон' ];
75  вернуть  DOKU_BASE.'lib / tpl /'.$ tpl . ' / ' ;
□□76}
77
78/**
79* Распечатать содержимое
80*
81* Эта функция используется для печати всего обычного контента.
82* (определяется глобальной переменной $ACT) путем вызова соответствующего
83* выходные функции из html.php
84*
85* Все, что не использует основной файл шаблона, не
86* обрабатывается этой функцией. ACL-списки здесь тоже не обрабатываются.
87*
88* @param  bool $ prependTOC следует ли здесь отображать оглавление?
89* @return  bool true, если есть какой-либо вывод
90*
91* @triggers TPL_ACT_RENDER
92* @triggers TPL_CONTENT_DISPLAY
93* @автор Андреас Гор <andi@splitbrain.org>
94*/
95функция  tpl_content ($ prependTOC = true )
96{
97  глобальный $ ACT ;
98  глобальная $ ИНФОРМАЦИЯ ;
99    $ INFO [ 'prependTOC' ] = $ prependTOC ;
100
101    ob_start ();
102  Событие :: createAndTrigger ( 'TPL_ACT_RENDER' , $ ACT ,
'tpl_content_core' );
103    $ html_output = ob_get_clean ();
104  Событие :: createAndTrigger ( 'TPL_CONTENT_DISPLAY' , $ html_output ,
function ($ html_output ) {
105        echo $ html_output ;
106    });
107
108  возврат ! пусто ($ html_output );
109}
```

```
110
111/**
112* Действие по умолчанию TPL_ACT_RENDER
113*
114* @return bool
115*/
116функция tpl_content_core ()
117{
118    $ router = ActionRouter :: getInstance ();
119  пытаться {
120    $ маршрутизатор -> getAction ()-> tplContent ();
121    } поймать ( FatalException $ e ) {
122        // не было контента для действия
123    сообщение ( hsc ($ e -> getMessage ()), -1 );
124    вернуть ложь ;
125    }
126  вернуть истину ;
127}
128
129/**
130* Размещает оглавление там, где вызывается функция
131*
132* Если вы используете это, вы, скорее всего, захотите вызвать tpl_content с помощью
133* ложный аргумент
134*
135* @param bool $ return Следует ли вернуть оглавление вместо его печати?
136* @возвращаемая строка
137*
138* @автор Андреас Гор <andi@splitbrain.org>
139*/
140функция tpl_toc ($ return = false )
141{
142  глобальный $ TOC ;
143  глобальный $ ACT ;
144  глобальный $ ID ;
145  глобальный $ REV ;
146  глобальная $ ИНФОРМАЦИЯ ;
147  глобальная $ conf ;
148    $ toc = [];
149
150  если ( is_array ($ TOC )) {
151        // если TOC был подготовлен в глобальном масштабе, всегда используйте его
152        $ toc = $ TOC ;
153    } elseif (($ ACT == 'show' || str_starts_with ($ ACT , 'export' )) &&
!$ REV && $ INFO [ 'exists' ]) {
154        // получить TOC из метаданных, отобразить при необходимости
155        $ meta = p_get_metadata ($ ID , '' , METADATA_RENDER_USING_CACHE
);
156        $ tocok = $ meta [ 'internal' ][ 'toc' ] ?? true ;
157        $ toc = $ meta [ 'description' ][ 'tableofcontents' ] ?? null ;
158    если (!$ tocok || ! is_array ($ toc ) || !$ conf [ 'tocminheads' ] ||
```

```
count ($ toc ) < $ conf [ 'tocminheads' ]) {
159             $ toc = [];
160         }
161     } elseif ($ ACT == 'admin' ) {
162         // попробуем загрузить оглавление плагина администратора
163         /** @var  AdminPlugin $plugin */
164     если ($ plugin = plugin_getRequestAdminPlugin ()) {
165             $ toc = $ plugin -> getTOC ();
166             $ TOC = $ toc ; // избежать последующей перестройки
167         }
168     }
169
170  Событие :: createAndTrigger ( 'TPL_TOC_RENDER' , $ toc , null , false
);
171     $ html = html_TOC ($ toc );
172  если ($ return ) вернуть $ html ;
173  эхо $ html ;
174  возвращаться '' ;
175}
176
177/**
178* Обработка содержимого страницы администратора
179*
180* @return  bool
181*
182* @автор Андреас Гор <andi@splitbrain.org>
183*/
184функция  tpl_admin ()
185{
186  глобальная $ ИНФОРМАЦИЯ ;
187  глобальный  $ TOC ;
188  глобальный $ ВХОД ;
189
190  $ плагин = null ;
191  $ класс = $ ВХОД -> стр ( 'страница' );
192  если (! пусто ($ класс )) {
193         $ pluginlist = plugin_list ( 'admin' );
194
195     если ( in_array ($ class , $ pluginlist )) {
196             // попытка загрузить плагин
197             /** @var  AdminPlugin $plugin */
198             $ plugin = plugin_load ( 'admin' , $ class );
199         }
200     }
201
202  если ($ plugin  instanceof  PluginInterface ) {
203         if (! is_array ($ TOC )) $ TOC = $ plugin -> getTOC (); //если TOC
еще не был запрошен
204     если ($ INFO [ 'prependTOC' ]) tpl_toc ();
205     $ плагин -> html ();
206     } еще {
```

```
    $ админ = новый Администратор ();
    $ админ -> показать ();
    }
  вернуть истину ;
}

/**
* Распечатайте правильные HTML-мета-заголовки
*
* Это необходимо разместить в заголовке вашего шаблона.
*
* @param  bool $ alt Нужно ли добавлять каналы и ссылки альтернативного формата?
* @return  bool
* @вызывает  JsonException
*
* @автор Андреас Гор <andi@splitbrain.org>
* @triggers TPL_METAHEADER_OUTPUT
*/
функция  tpl_metaheaders ($ alt = true )
{
  глобальный $ ID ;
  глобальный $ REV ;
  глобальная $ ИНФОРМАЦИЯ ;
  глобальный $ JSINFO ;
  глобальный $ ACT ;
  глобальный $ ЗАПРОС ;
  глобальный $ lang ;
  глобальная $ conf ;
  глобальная $ updateVersion ;
    /** @var  Вход $INPUT */
  глобальный $ ВХОД ;

    // подготавливаем массив головок
  $ голова = [];

    // подготовить seed для js и css
    $ tseed = $ updateVersion ;
  $ зависит = getConfigFiles ( 'main' );
  $ зависит [] = DOKU_CONF . "tpl/" . $ conf [ 'template' ] . "/style.ini" ;
    foreach ($ зависит от $ f ) $ tseed .= @ filemtime ($ f );
    $ tseed = md5 ($ tseed );

    // обычные вещи
    $ head [ 'meta' ][] = [ 'name' => 'generator' , 'content' => 'DokuWiki' ];
  если ( действиеOK ( 'поиск' )) {
        $ head [ 'ссылка' ][] = [
            'rel' => 'поиск' ,
            'тип' => 'приложение / opensearchdescription +xml' ,
```

```
255             'href' => DOKU_BASE . ' lib / exe / opensearch.php ' ,
256             'заголовок' => $ conf [ 'заголовок' ]
257         ];
258     }
259 
260     $ head [ 'link' ][] = [ 'rel' => 'start' , 'href' => DOKU_BASE ];
261   если ( действиеOK ( 'индекс' )) {
262         $ head [ 'ссылка' ][] = [
263             'rel' => 'содержимое' ,
264             'href' => wl ($ ID , 'do=index' , false , '&' ),
265             'title' => $ lang [ 'btn_index' ]
266         ];
267     }
268 
269   если ( действиеOK ( 'манифест' )) {
270         $ head [ 'ссылка' ][] = [
271             'rel' => 'манифест' ,
272             'href' => DOKU_BASE . ' lib / exe / manifest.php '
273         ];
274     }
275 
276     $ styleUtil = new  StyleUtils ();
277     $ styleIni = $ styleUtil -> cssStyleini ();
278     $ replacements = $ styleIni [ 'replacements' ];
279   если (! пусто ($ replacements [ '__theme_color__' ])) {
280     $ голова [ 'мета' ][] = [
281             'имя' => 'цвет темы' ,
282             'content' => $ replacements [ '__theme_color__' ]
283         ];
284     }
285 
286   если ($ альт ) {
287     если ( действиеOK ( 'rss' )) {
288             $ head [ 'ссылка' ][] = [
289                 'rel' => 'альтернативный' ,
290                 'тип' => 'приложение / rss +xml' ,
291                 'title' => $ lang [ 'btn_recent' ],
292                 'href' => DOKU_BASE . ' feed.php '
293             ];
294             $ head [ 'ссылка' ][] = [
295                 'rel' => 'альтернативный' ,
296                 'тип' => 'приложение / rss +xml' ,
297                 'title' => $ lang [ 'currentns' ],
298                 'href' => DOKU_BASE . ' feed.php ?mode=list&ns=' . (
isset ($ INFO ) ? $ INFO [ 'namespace' ] : '' )
299             ];
300         }
301     если (($ ACT == 'показать' || $ ACT == 'поиск' ) && $ INFO [ 'записываемый'
]) {
302             $ head [ 'ссылка' ][] = [
303                 'rel' => 'редактировать' ,
```

```
304                 'title' => $ lang [ 'btn_edit' ],
305                 'href' => wl ($ ID , 'do=edit' , false , '&' )
306             ];
307         }
308 
309     если ( actionOK ( 'rss' ) && $ ACT == 'search' ) {
310             $ head [ 'ссылка' ][] = [
311                 'rel' => 'альтернативный' ,
312                 'тип' => 'приложение / rss +xml' ,
313                 'title' => $ lang [ 'searchresult' ],
314                 'href' => DOKU_BASE . ' feed.php ?mode=search&q=' . $
ЗАПРОС
315             ];
316         }
317 
318     если ( actionOK ( 'export_xhtml' )) {
319             $ head [ 'ссылка' ][] = [
320                 'rel' => 'альтернативный' ,
321                 'тип' => 'текст / html ' ,
322                 'title' => $ lang [ 'plainhtml' ],
323                 'href' => экспортссылка ($ ID , 'xhtml' , '' , false , '&'
)
324             ];
325         }
326 
327     если ( actionOK ( 'export_raw' )) {
328             $ head [ 'ссылка' ][] = [
329                 'rel' => 'альтернативный' ,
330                 'тип' => 'текст / обычный ' ,
331                 'title' => $ lang [ 'wikimarkup' ],
332                 'href' => экспортссылка ($ ID , 'raw' , '' , false , '&' )
333             ];
334         }
335     }
336 
337     // настройка тегов робота, подходящих для разных режимов
338   если (($ ACT == 'show' || $ ACT == 'export_xhtml' ) && !$ REV ) {
339     если ($ INFO [ 'существует' ]) {
340             //задержка индексации:
341       если (( время () - $ INFO [ 'lastmod' ]) >= $ conf [ 'indexdelay' ]
&& ! isHiddenPage ($ ID )) {
342                 $ head [ 'meta' ][] = [ 'name' => 'robots' , 'content' =>
'index,follow' ];
343             } еще {
344                 $ head [ 'meta' ][] = [ 'name' => 'robots' , 'content' =>
'noindex,nofollow' ];
345             }
346             $ canonicalUrl = wl ($ ID , '' , true , '&' );
347       если ($ ID == $ conf [ 'start' ]) {
348                 $ canonicalUrl = DOKU_URL ;
349             }
```

```
350             $ head [ 'link' ][] = [ 'rel' => 'canonical' , 'href' => $
canonicalUrl ];
351         } еще {
352             $ head [ 'meta' ][] = [ 'name' => 'robots' , 'content' =>
'noindex,follow' ];
353         }
354     } elseif ( определено ('DOKU_MEDIADETAIL')) {
355          $ head [ 'meta' ][] = [ 'name' => 'robots' , 'content' =>
'index,follow' ];
356     } еще {
357         $ head [ 'meta' ][] = [ 'name' => 'robots' , 'content' =>
'noindex,nofollow' ];
358     }
359
360     // установить метаданные
361   если ($ ACT == 'показать' || $ ACT == 'export_xhtml' ) {
362         // ключевые слова (явные или неявные)
363     если (! пусто ($ INFO [ 'meta' ][ 'subject' ])) {
364             $ head [ 'meta' ][] = [ 'name' => 'keywords' , 'content' =>
implode ( ',' , $ INFO [ 'meta' ][ 'subject' ])];
365         } еще {
366             $ head [ 'meta' ][] = [ 'name' => 'keywords' , 'content' =>
str_replace ( ':' , ',' , $ ID )];
367         }
368     }
369
370     // загрузка таблиц стилей
371     $ head [ 'ссылка' ][] = [
372         'rel' => 'таблица стилей' ,
373         'href' => DOKU_BASE . ' lib / exe / css.php ?t=' . rawurlencode
($ conf [ 'template' ]) . '&tseed=' . $ tseed
374     ];
375
376     $ script = "var NS=" . ( isset ($ INFO ) ? $ INFO [ 'namespace' ] :
'' ) . "';" ;
377   если ($ conf [ 'useacl' ] && $ INPUT -> сервер -> str ( 'REMOTE_USER' ))
{
378     $ скрипт .= "var SIG=" . тулбар_сигнатура () . ";" ;
379     }
380     jsinfo ();
381     $ script .= 'var JSINFO = ' . json_encode ($ JSINFO ,
JSON_THROW_ON_ERROR ) . ';' ;
382     $ script .= '(function(H){H.className=H.className.replace(/\bno-
js\b/, \' js \' )})(document.documentElement);' ;
383     $ head [ 'script' ][] = [ '_data' => $ script ];
384
385     // загрузить jquery
386     $ jquery = getCdnUrls ();
387     foreach ($ jquery  как $ src ) {
388         $ head [ 'скрипт' ][] = [
389                 '_data' => '' ,
```

```
390                 'источник' => $ источник
391             ] + ($ conf [ 'defer_js' ] ? [ 'defer' => 'defer' ] : []);
392     }
393
394     // загружаем наш диспетчер javascript
395     $ head [ 'скрипт' ][] = [
396             '_data' => '' ,
397             'src' => DOKU_BASE . ' lib / exe / js.php ' . '?t=' .
rawurlencode ($ conf [ 'шаблон' ]) . '&tseed=' . $ семя
398         ] + ($ conf [ 'defer_js' ] ? [ 'defer' => 'defer' ] : []);
399
400     // вызвать событие здесь
401   Событие :: createAndTrigger ( 'TPL_METAHEADER_OUTPUT' , $ head ,
'_tpl_metaheaders_action' , true );
402   вернуть истину ;
403}
404
405/**
406* печатает массив, созданный tpl_metaheaders
407*
408* $data — это массив различных тегов заголовков. Каждый тег может иметь несколько
409* экземпляры. Атрибуты задаются как пары ключ-значение. Значения будут HTML
410* кодируются автоматически, поэтому их следует предоставлять как есть в массиве
$data.
411*
412* Для тегов, имеющих атрибут body, укажите данные body в специальном поле
413* атрибут '_data'. Это поле НЕ БУДЕТ ЭКРАНИРОВАНО автоматически.
414*
415* Встроенные скрипты будут использовать любой одноразовый номер, указанный в
переменной среды «NONCE».
416*
417* @param   массив $ данные
418*
419* @автор Андреас Гор <andi@splitbrain.org>
420*/
421функция  _tpl_metaheaders_action ($ data )
422{
423     $ nonce = getenv ( 'NONCE' );
424     foreach ($ data  как $ tag => $ inst ) {
425         foreach ($ inst  as $ attr ) {
426       если ( пусто ($ attr )) {
427         продолжать ;
428             }
429       если ($ nonce && $ tag == 'script' && ! empty ($ attr [ '_data' ]))
{
430                 $ attr [ 'nonce' ] = $ nonce ; // добавить nonce к
встроенным тегам скрипта
431             }
432             echo  '<' , $ tag , ' ' , buildAttributes ($ attr );
433       если ( isset ($ attr [ '_data' ]) || $ tag == 'script' ) {
434                 echo  '>' , $ attr [ '_data' ] ?? '' , '</' , $ tag , '>'
```

```
;
435             } еще {
436         эхо '/>' ;
437             }
438       эхо " \n " ;
439         }
440     }
441}
442
443/**
444* Вывести данный скрипт как встроенный тег скрипта
445*
446* Эта функция добавит атрибут nonce, если он доступен.
447*
448* Скрипт НЕ экранируется автоматически!
449*
450* @param  string $ скрипт
451* @param  bool $ return Возврат или прямая печать?
452* @return  string | недействительный
453*/
454функция  tpl_inlineScript ($ script , $ return = false )
455{
456     $ nonce = getenv ( 'NONCE' );
457   если ($ nonce ) {
458     $ скрипт = '<script nonce="' . $ nonce . '">' . $ скрипт . '</script>' ;
459     } еще {
460     $ скрипт = '<скрипт>' . $ скрипт . '</скрипт>' ;
461     }
462
463   если ($ return ) return $ script ;
464   эхо $ скрипт ;
465}
466
467/**
468* Распечатать ссылку
469*
470* Просто создает ссылку.
471*
472* @param  string $ url
473* @param  string $ имя
474* @param  string $ еще
475* @param  bool $ return если true вернуть ссылку html, в противном случае
вывести
476* @return  bool | строка html ссылки или true, если выводится
477*
478* @автор Андреас Гор <andi@splitbrain.org>
479*/
480функция  tpl_link ($ url , $ name , $ more = '' , $ return = false )
481{
482     $ out = '<a href="' . $ url . '" ' ;
483   если ($ more ) $ out .= ' ' . $ more ;
```

```
484     $ out .= "> $ имя </a>" ;
485   если ($ return ) вернуть $ out ;
486   вывести $ ;
487   вернуть истину ;
488 }
489
490 /**
491  * Печатает ссылку на WikiPage
492  *
493  * Обертка вокруг html_wikilink
494  *
495  * @param  string $ id идентификатор страницы
496  * @param  string | null $ name имя ссылки
497  * @param  bool $ возврат
498  * @return  true | строка
499  *
500  * @автор Андреас Гор <andi@splitbrain.org>
501  */
502 функция  tpl_pagelink ($ id , $ name = null , $ return = false )
503 {
504     $ out = '<bdi>' . html_wikilink ( $ id , $ name ) . '</bdi>' ;
505   если ($ return ) вернуть $ out ;
506   вывести $ ;
507   вернуть истину ;
508 }
509
510 /**
511  * получить родительскую страницу
512  *
513  * Пытается выяснить, какая страница является родительской.
514  * возвращает false, если ничего не доступно
515  *
516  * @param  string $ id идентификатор страницы
517  * @return  false | строка
518  *
519  * @автор Андреас Гор <andi@splitbrain.org>
520  */
521 функция  tpl_getparent ($ id )
522 {
523     $ resolver = new  PageResolver ( 'root' );
524
525     $ parent = getNS ($ id ). ':' ;
526     $ parent = $ resolver -> resolveId ($ parent );
527   если ($ родитель == $ идентификатор ) {
528         $ pos = strrpos ( getNS ($ id ), ':' );
529         $ parent = substr ($ parent , 0 , $ pos ). ':' ;
530         $ parent = $ resolver -> resolveId ($ parent );
531     если ($ parent == $ id ) вернуть false ;
532     }
533   вернуть $ родитель ;
534 }
```

```
535
536/**
537* Распечатать одну из кнопок
538*
539* @param  string $ тип
540* @param  bool $ возврат
541* @return  bool | string html, или false, если данных нет, true, если
выведено
542* @see     tpl_get_action
543*
544* @автор Адриан Лэнг <mail@adrianlang.de>
545* @deprecated 2017-09-01 см. devel:menus
546*/
547функция  tpl_button ($ тип , $ возврат = ложь )
548{
549    dbg_deprecated ( 'см. devel:menus' );
550    $ data = tpl_get_action ($ type );
551  если ($ данные === ложь ) {
552    вернуть ложь ;
553    } elseif (! is_array ($ data )) {
554        $ out = sprintf ($ data , 'button' );
555    } еще {
556        /**
557         * @var  string $accesskey
558         * @var  string $id
559         * @var  string $метод
560         * @var  массив $params
561         */
562    извлечь ($ data );
563    если ($ id === '#dokuwiki__top' ) {
564            $ out = html_topbtn ();
565        } еще {
566            $ out = html_btn ($ type , $ id , $ accesskey , $ params , $
method );
567        }
568    }
569  если ($ return ) вернуть $ out ;
570  вывести $ ;
571  вернуть истину ;
572}
573
574/**
575* Как кнопки действий, но ссылки
576*
577* @param  string $ тип действие команда
578* @param  string $ pre префикс ссылки
579* @param  string $ suf суффикс ссылки
580* @param  string $ внутренний innerHML ссылки
581* @param  bool $ return если true, то возвращает html, в противном случае
печатает
582* @return  bool | string html или false, если данных нет, true, если
```

```
выведено
583*
584* @see     tpl_get_action
585* @автор Адриан Лэнг <mail@adrianlang.de>
586* @deprecated 2017-09-01 см. devel:menus
587*/
588функция  tpl_actionlink ($ type , $ pre = '' , $ suf = '' , $ inner = ''
, $ return = false )
589{
590    dbg_deprecated ( 'см. devel:menus' );
591  глобальный $ lang ;
592    $ data = tpl_get_action ($ type );
593  если ($ данные === ложь ) {
594    вернуть ложь ;
595    } elseif (! is_array ($ data )) {
596        $ out = sprintf ($ data , 'link' );
597    } еще {
598        /**
599         * @var  string $accesskey
600         * @var  string $id
601         * @var  string $метод
602         * @var  bool $nofollow
603         * @var  массив $params
604         * @var  string $replacement
605         */
606    извлечь ($ data );
607    если ( strpos ($ id , '#' ) === 0 ) {
608            $ linktarget = $ id ;
609        } еще {
610            $ linktarget = wl ($ id , $ params );
611        }
612        $ caption = $ lang [ 'btn_' . $ type ];
613    если ( strpos ($ caption , '%s' )) {
614            $ caption = sprintf ($ caption , $ replacement );
615        }
616        $ akey = '' ;
617        $ addTitle = '' ;
618    если ($ accesskey ) {
619            $ akey = 'accesskey="' . $ accesskey . '" ' ;
620            $ addTitle = '[' . strtoupper ($ accesskey ) . ']' ;
621        }
622        $ rel = $ nofollow ? 'rel="nofollow" ' : '' ;
623        $ out = tpl_link (
624      $ ссылкацель ,
625            $ pre . ($ inner ?: $ caption ) . $ suf ,
626            'класс="действие' . $ тип . '" ' .
627            $ akey . $ rel .
628            'title="' . hsc ($ caption ) . $ addTitle . '"' ,
629      истинный
630        );
631    }
```

```
632  если ($ return ) вернуть $ out ;
633  вывести $ ;
634   вернуть истину ;
635}
636
637/**
638* Проверьте действия и получите данные для кнопок и ссылок
639*
640* @param  string $ тип
641* @return  массив | bool | строка
642*
643* @автор Адриан Лэнг <mail@adrianlang.de>
644* @автор Андреас Гор <andi@splitbrain.org>
645* @автор Маттиас Гримм <matthiasgrimm@users.sourceforge.net>
646* @deprecated 2017-09-01 см. devel:menus
647*/
648функция  tpl_get_action ($ тип )
649{
650    dbg_deprecated ( 'см. devel:menus' );
651  если ($ type == 'history' ) $ type = 'revisions' ;
652  если ($ type == 'подписка' ) $ type = 'подписка' ;
653  если ($ type == 'img_backto' ) $ type = 'imgBackto' ;
654
655    $ class = ' \\ dokuwiki \\ Меню \\ Элемент \\' . ucfirst ($ type );
656  если ( class_exists ($ class )) {
657    пытаться {
658            /** @var  AbstractItem $item */
659            $ item = новый $ class ();
660            $ data = $ item -> getLegacyData ();
661      $ неизвестно = ложь ;
662        } catch ( RuntimeException $ игнорируется ) {
663      вернуть ложь ;
664        }
665    } еще {
666    глобальный $ ID ;
667    $ данные = [
668            'accesskey' => null ,
669            'тип' => $ тип ,
670            'id' => $ ID ,
671            'метод' => 'получить' ,
672            'params' => [ 'do' => $ type ],
673            'nofollow' => правда ,
674            'замена' => ''
675        ];
676    $ неизвестно = правда ;
677    }
678
679    $ evt = новое событие ( 'TPL_ACTION_GET' , $ data );
680  если ($ evt -> advice_before ()) {
681        //обработка неизвестных типов
682    если ($ неизвестно ) {
```

```
683             $ data = '[неизвестный тип %s]' ;
684         }
685     }
686     $ evt -> advice_after ();
687   снято ($ evt );
688 
689   вернуть $ данные ;
690}
691 
692/**
693* Обертка вокруг tpl_button() и tpl_actionlink()
694*
695* @param  string $ тип действие команда
696* @param  bool $ ссылка ссылка или кнопка формы?
697* @param  string | bool $ wrapper Обертка HTML-элемента
698* @param  bool $ return return или print
699* @param  string $ pre префикс для ссылок
700* @param  string $ suf суффикс для ссылок
701* @param  string $ внутренний внутренний HTML для ссылок
702* @return  bool | строка
703*
704* @автор Аника Хенке <anika@selfthinker.org>
705* @deprecated 2017-09-01 см. devel:menus
706*/
707функция  tpl_action ($ type , $ link = false , $ wrapper = false , $
return = false , $ pre = '' , $ suf = '' , $ inner = '' )
708{
709     dbg_deprecated ( 'см. devel:menus' );
710     $ out = '' ;
711   если ($ ссылка ) {
712         $ out .= tpl_actionlink ($ type , $ pre , $ suf , $ inner , true
);
713     } еще {
714         $ out .= tpl_button ( $ type , true );
715     }
716   если ($ out && $ wrapper ) $ out = "< $ wrapper > $ out </ $ wrapper >"
;
717 
718   если ($ return ) вернуть $ out ;
719   вывести $ ;
720     return ( bool )$ out ;
721}
722 
723/**
724* Распечатать форму поиска
725*
726* Если первый параметр задан как div с идентификатором 'qsearch_out', то будет
727* добавляется, который инструктирует страницу ajax quicksearch включиться и
разместить
728* его вывод в этот div. Второй параметр управляет собственным
729* атрибут автозаполнения. Если установлено значение false, этот атрибут будет
```

```
установлен с
730* значение "off" указывает браузеру отключить встроенные функции
731* функция автодополнения (MSIE и Firefox)
732*
733* @param  bool $ ajax
734* @param  bool $ автозаполнение
735* @return  bool
736*
737* @автор Андреас Гор <andi@splitbrain.org>
738*/
739функция  tpl_searchform ($ ajax = true , $ autocomplete = true )
740{
741  глобальный $ lang ;
742  глобальный $ ACT ;
743  глобальный $ ЗАПРОС ;
744  глобальный $ ID ;
745
746    // не печатать форму поиска, если действие поиска отключено
747  если (! actionOK ( 'search' )) вернуть false ;
748
749    $ searchForm = новая форма ([
750        'действие' => wl (),
751        'метод' => 'получить' ,
752        'роль' => 'поиск' ,
753        'класс' => 'поиск' ,
754        'id' => 'dw__search' ,
755    ], истинный );
756    $ searchForm -> addTagOpen ( 'div' )-> addClass ( 'no' );
757    $ searchForm -> setHiddenField ( 'сделать' , 'поиск' );
758    $ searchForm -> setHiddenField ( 'id' , $ ID );
759    $ searchForm -> addTextInput ( 'q' )
760        -> добавитьКласс ( 'редактировать' )
761        -> атрибуты ([
762            'заголовок' => '[F]' ,
763            'accesskey' => 'f' ,
764            'placeholder' => $ lang [ 'btn_search' ],
765            'autocomplete' => $ autocomplete ? 'on' : 'off' ,
766        ])
767        -> идентификатор ( 'qsearch__in' )
768        -> val ($ ACT === 'поиск' ? $ QUERY : '' )
769        -> useInput ( false );
770    $ searchForm -> addButton ( '' , $ lang [ 'btn_search' ])-> attrs ([
771        'тип' => 'отправить' ,
772        'title' => $ lang [ 'btn_search' ],
773    ]);
774  если ($ ajax ) {
775        $ searchForm -> addTagOpen ( 'div' )-> id ( 'qsearch__out' )->
addClass ( 'ajax_qsearch JSpopup' );
776        $ searchForm -> addTagClose ( 'div' );
777    }
778    $ searchForm -> addTagClose ( 'div' );
```

```
779
780     echo $ searchForm -> toHTML ( 'Быстрый поиск' );
781
782   вернуть истину ;
783}
784
785/**
786* Распечатать след навигационной цепочки
787*
788* @param  string $ sep Разделитель между записями
789* @param  bool $ return return или print
790* @return  bool | строка
791*
792* @автор Андреас Гор <andi@splitbrain.org>
793*/
794функция  tpl_breadcrumbs ($ sep = null , $ return = false )
795{
796   глобальный $ lang ;
797   глобальная $ conf ;
798
799     //проверить, включено ли
800   если (!$ conf [ 'хлебные крошки' ]) вернуть false ;
801
802     //установить значение по умолчанию
803   если ( is_null ($ sep )) $ sep = '•' ;
804
805     $ out = '' ;
806
807     $ crumbs = breadcrumbs (); //настройка трассировки крошек
808
809     $ crumbs_sep = ' <span class="bcsep">' . $ sep . '</span> ' ;
810
811     //рендерим крошки, выделяем последнюю
812     $ out .= '<span class="bchead">' . $ lang [ 'хлебные крошки' ] .
'</span>' ;
813     $ last = count ($ crubs );
814   $ я = 0 ;
815     foreach ($ крошки как $ id => $ name ) {
816     $ я ++;
817         $ out .= $ crumbs_sep ;
818     если ($ i == $ last ) $ out .= '<span class="curid">' ;
819         $ out .= '<bdi>' . tpl_link ( wl ( $ id ), hsc ( $ name ), '
class="breadcrumbs" title="' . $ id . '"' , true ) . '</bdi>' ;
820     если ($ i == $ last ) $ out .= '</span>' ;
821     }
822   если ($ return ) вернуть $ out ;
823   вывести $ ;
824     return ( bool )$ out ;
825}
826
827/**
```

```
828* Иерархическая навигационная цепочка
829*
830* Этот код был предложен в качестве замены обычным хлебным крошкам.
831* Имеет смысл только при наличии глубокой структуры сайта.
832*
833* @param  string $ sep Разделитель между записями
834* @param  bool $ return return или print
835* @return  bool | строка
836*
837* @todo     может вести себя странно в языках с письмом справа налево
838* @автор <fredrik@averpil.com>
839* @автор Андреас Гор <andi@splitbrain.org>
840* @автор Найджел Макни <oracle.shinoda@gmail.com>
841* @автор Шон Коутс <sean@caedmon.net>
842*/
843функция  tpl_youarehere ($ sep = null , $ return = false )
844{
845  глобальная $ conf ;
846  глобальный $ ID ;
847  глобальный $ lang ;
848
849    // проверить, включено ли
850  если (!$ conf [ 'youarehere' ]) вернуть false ;
851
852    //установить значение по умолчанию
853  если ( is_null ($ sep )) $ sep = ' » ' ;
854
855    $ out = '' ;
856
857    $ parts = Explode ( ':' , $ ID );
858    $ count = count ($ parts );
859
860    $ out .= '<span class="bchead">' . $ lang [ 'выздесь' ] . ' </span>' ;
861
862    // всегда печатать стартовую страницу
863    $ out .= '<span class="home">' . tpl_pagelink ( ':' . $ conf [
'start' ], null , true ) . '</span>' ;
864
865    // распечатать промежуточные ссылки пространства имен
866  $ часть = '' ;
867  для ($ i = 0 ; $ i < $ count - 1 ; $ i ++) {
868    $ часть .= $ часть [$ i ]. ':' ;
869    $ страница = $ часть ;
870        if ($ page == $ conf [ 'start' ]) continue ; // Пропустить начальную
страницу
871
872        // выход
873        $ out .= $ sep . tpl_pagelink ( $ page , null , true );
874    }
875
876    // распечатать текущую страницу, пропустив начальную страницу, пропустив индекс
```

```
пространства имен
877  если ( isset ($ страница )) {
878        $ page = ( new  PageResolver ( 'root' ))-> resolveId ( $ page );
879    если ($ страница == $ часть . $ части [$ я ]) {
880      если ($ return ) вернуть $ out ;
881      вывести $ ;
882       вернуть истину ;
883        }
884    }
885  $ страница = $ часть . $ части [$ i ];
886  если ($ страница == $ конф [ 'старт' ]) {
887    если ($ return ) вернуть $ out ;
888    вывести $ ;
889     вернуть истину ;
890    }
891    $ out .= $ sep ;
892    $ out .= tpl_pagelink ($ page , null , true );
893  если ($ return ) вернуть $ out ;
894  вывести $ ;
895     return ( bool )$ out ;
896}
897
898/**
899* Распечатать информацию, если пользователь вошел в систему
900* и в этом случае показывать полное имя
901*
902* Можно ли в будущем добавить ссылку на профиль?
903*
904* @return  bool
905*
906* @автор Андреас Гор <andi@splitbrain.org>
907*/
908функция  tpl_userinfo ()
909{
910  глобальный $ lang ;
911    /** @var  Вход $INPUT */
912  глобальный $ ВХОД ;
913
914  если ($ INPUT -> сервер -> str ( 'REMOTE_USER' )) {
915        echo $ lang [ 'loggedinas' ] . ' ' . userlink ();
916    вернуть истину ;
917    }
918  вернуть ложь ;
919}
920
921/**
922* Распечатать некоторую информацию о текущей странице
923*
924* @param  bool $ ret возвращает содержимое вместо его печати
925* @return  bool | строка
926*
```

```
927* @автор Андреас Гор <andi@splitbrain.org>
928*/
929функция  tpl_pageinfo ($ ret = false )
930{
931  глобальная $ conf ;
932  глобальный $ lang ;
933  глобальная $ ИНФОРМАЦИЯ ;
934  глобальный $ ID ;
935
936    // возвращаем, если нам не разрешено просматривать страницу
937  если (! auth_quickaclcheck ($ ID )) {
938    вернуть ложь ;
939    }
940
941    // подготовить дату и путь
942    $ fn = $ INFO [ 'путь_к_файлу' ];
943  если (!$ conf [ 'полный_путь' ]) {
944    если ($ ИНФОРМАЦИЯ [ 'рев' ]) {
945            $ fn = str_replace ($ conf [ 'olddir' ] . '/' , '' , $ fn );
946        } еще {
947            $ fn = str_replace ($ conf [ 'datadir' ] . '/' , '' , $ fn );
948        }
949    }
950    $ fn = utf8_decodeFN ($ fn );
951    $ date = dformat ($ INFO [ 'lastmod' ]);
952
953    // распечатать это
954  если ($ INFO [ 'существует' ]) {
955        $ out = '<bdi>' . $ fn . '</bdi>' ;
956        $ out .= ' · ' ;
957        $ out .= $ lang [ 'lastmod' ];
958        $ out .= ' ' ;
959        $ out .= $ date ;
960    если ($ INFO [ 'редактор' ]) {
961            $ out .= ' ' . $ lang [ 'by' ] . ' ' ;
962            $ out .= '<bdi>' . editorinfo ($ INFO [ 'editor' ]) .
'</bdi>' ;
963        } еще {
964            $ out .= ' (' . $ lang [ 'external_edit' ] . ')' ;
965        }
966    если ($ INFO [ 'заблокировано' ]) {
967            $ out .= ' · ' ;
968            $ out .= $ lang [ 'lockedby' ];
969            $ out .= ' ' ;
970            $ out .= '<bdi>' . editorinfo ($ INFO [ 'locked' ]) .
'</bdi>' ;
971        }
972    если ($ рет ) {
973      возврат $ из ;
974        } еще {
975      вывести $ ;
```

```
976       вернуть истину ;
977         }
978     }
979   вернуть ложь ;
980}
981
982/**
983* Печатает или возвращает имя указанной страницы (текущей, если не указано).
984*
985* Если включено использование заголовка, будет использоваться первый заголовок, в противном случае
986* используется указанный идентификатор.
987*
988* @param  string $ id идентификатор страницы
989* @param  bool $ ret возвращает содержимое вместо печати
990* @return  bool | строка
991*
992* @автор Андреас Гор <andi@splitbrain.org>
993*/
994функция  tpl_pagetitle ($ id = null , $ ret = false )
995{
996   глобальные $ ACT , $ conf , $ lang ;
997
998   если ( is_null ($ id )) {
999     глобальный $ ID ;
1000         $ id = $ ID ;
1001     }
1002
1003   $ имя = $ идентификатор ;
1004   если ( useHeading ( 'навигация' )) {
1005         $ first_heading = p_get_first_heading ($ id );
1006     если ($ first_heading ) $ name = $ first_heading ;
1007     }
1008
1009     // заголовок страницы по умолчанию — это имя страницы, измените его с помощью текущего действия
1010   переключатель ($ ACT ) {
1011         // административные функции
1012     случай «администратор» :
1013             $ page_title = $ lang [ 'btn_admin' ];
1014             // попробуем получить имя плагина
1015             /** @var  AdminPlugin $plugin */
1016       если ($ plugin = plugin_getRequestAdminPlugin ()) {
1017                 $ plugin_title = $ plugin -> getMenuText ($ conf [ 'lang' ]);
1018                 $ page_title = $ plugin_title ?: $ plugin -> getPluginName ();
1019             }
1020       перерыв ;
1021
1022         // показать действие как заголовок
```

```
1023     случай «логин» :
1024     кейс «профиль» :
1025     случай «регистр» :
1026     случай 'resendpwd' :
1027     случай «индекс» :
1028     случай «поиск» :
1029               $ page_title = $ lang [ 'btn_' . $ ACT ];
1030       перерыв ;
1031
1032          // добавить ручку во время редактирования
1033     случай «редактирование» :
1034     случай «предварительный просмотр» :
1035               $ page_title = "✎ " . $ name ;
1036       перерыв ;
1037
1038          // добавить действие к названию страницы
1039     дело «пересмотры» :
1040               $ page_title = $ name . ' - ' . $ lang [ 'btn_revs' ];
1041       перерыв ;
1042
1043          // добавить действие к названию страницы
1044     случай «обратная ссылка» :
1045     случай «недавний» :
1046     случай «подписаться» :
1047               $ page_title = $ name .'- ' .$ lang [ 'btn_' .$ ACT ];
1048       перерыв ;
1049
1050     по умолчанию : // SHOW и все остальное, что не включено
1051               $ page_title = $ name ;
1052     }
1053
1054   если ($ рет ) {
1055     вернуть  hsc ($ page_title );
1056     } еще {
1057          echo  hsc ($ page_title );
1058     вернуть истину ;
1059     }
1060}
1061
1062/**
1063* Возвращает запрошенный тег EXIF / IPTC из текущего изображения
1064*
1065* Если $tags — это массив, то все заданные теги проверяются до тех пор, пока не
будет найден
1066* значение найдено. Если значение не найдено, возвращается $alt.
1067*
1068* Какие тексты известны, определяется в функциях _exifTagNames
1069* и _iptcTagNames() в inc / jpeg.php (Вам необходимо добавить IPTC
1070* к именам последнего)
1071*
1072* Разрешено только в: detail.php
```

```
1073*
1074* @param  array | string $ tags  тег или массив тегов для проверки
1075* @param  string $ alt  альтернативный вывод, если данные не найдены
1076* @param  null | string $ src  источник изображения, если не указан,
используется глобальный  $SRC
1077* @возвращаемая  строка
1078*
1079* @автор Андреас Гор  <andi@splitbrain.org>
1080*/
1081функция  tpl_img_getTag ($ tags , $ alt = '' , $ src = null )
1082{
1083    // Инициализация  Exif-ридера
1084  глобальный $ SRC , $ imgMeta ;
1085
1086  если ( is_null ($ src )) $ src = $ SRC ;
1087  если ( is_null ($ src )) вернуть $ alt ;
1088
1089  если (! isset ($ imgMeta )) {
1090        $ imgMeta = new  JpegMeta ($ src );
1091    }
1092  если ($ imgMeta === false ) вернуть $ alt ;
1093    $ info = cleanText ($ imgMeta -> getField ($ tags ));
1094  если (!$ info ) вернуть $ alt ;
1095  вернуть $ информацию ;
1096}
1097
1098
1099/**
1100* Мусор собирает открытый объект  JpegMeta.
1101*/
1102функция  tpl_img_close ()
1103{
1104  глобальный $ imgMeta ;
1105    $ imgMeta = null ;
1106}
1107
1108/**
1109* Выводит  HTML-список описаний метатегов текущего изображения
1110*/
1111функция  tpl_img_meta ()
1112{
1113  глобальный $ lang ;
1114
1115  $ теги = tpl_get_img_meta ();
1116
1117  эхо '<dl>' ;
1118    foreach ($ теги  как $ теги ) {
1119        $ label = $ lang [$ tag [ 'langkey' ]];
1120    если (!$ label ) $ label = $ tag [ 'langkey' ]. ':' ;
1121
1122        echo  '<dt>' . $ label . '</dt><dd>' ;
```

```
1123     если ($ тег [ 'тип' ] == 'дата' ) {
1124             echo  dformat ($ тег [ 'значение' ]);
1125         } еще {
1126             echo  hsc ($ тег [ 'значение' ]);
1127         }
1128     эхо '</dd>' ;
1129     }
1130  эхо '</dl>' ;
1131}
1132
1133/**
1134* Возвращает метаданные, настроенные в файле конфигурации mediameta, готовые для создания html
1135*
1136* @return   массив с массивами, содержащими записи:
1137* - строка langkey key для поиска в переменной $lang, если не найдена, выводится как есть
1138* - строковый тип значения
1139* - строковое значение тега (неэкранированное)
1140*/
1141функция  tpl_get_img_meta ()
1142{
1143
1144     $ config_files = getConfigFiles ( 'mediameta' );
1145     foreach ($ config_files  как $ config_file ) {
1146     если ( file_exists ($ config_file )) {
1147       включить ($ config_file );
1148         }
1149     }
1150  $ теги = [];
1151     foreach ($ поля  как $ тег ) {
1152     $ т = [];
1153     если (! пусто ($ тег [ 0 ])) {
1154       $ т = [$ тег [ 0 ]];
1155         }
1156     если ( isset ($ тег [ 3 ]) && is_array ($ тег [ 3 ])) {
1157             $ t = array_merge ($ t , $ tag [ 3 ]);
1158         }
1159         $ value = tpl_img_getTag ($ t );
1160     если ($ значение ) {
1161       $ теги [] = [ 'langkey' => $ тег [ 1 ], 'type' => $ тег [ 2 ], 'value' => $ значение ];
1162         }
1163     }
1164  вернуть $ теги ;
1165}
1166
1167/**
1168* Печатает изображение со ссылкой на полноразмерную версию
1169*
1170* Разрешено только в: detail.php
```

```
1171*
1172* @triggers TPL_IMG_DISPLAY
1173* @param  int $ maxwidth - максимальная ширина изображения
1174* @param  int $ maxheight - максимальная высота изображения
1175* @param  bool $ link - ссылка на исходный размер?
1176* @param  array $ params - дополнительные атрибуты изображения
1177* @return  bool Результат TPL_IMG_DISPLAY
1178*/
1179функция  tpl_img ($ maxwidth = 0 , $ maxheight = 0 , $ link = true , $
params = null )
1180{
1181  глобальный $ IMG ;
1182    /** @var  Вход $INPUT */
1183  глобальный $ ВХОД ;
1184  глобальный $ REV ;
1185    $ w = ( int ) tpl_img_getTag ( 'Файл.Ширина' );
1186    $ h = ( int ) tpl_img_getTag ( 'Файл.Высота' );
1187
1188    //изменить размер до указанных максимальных значений
1189Коэффициент   $ = 1 ;
1190  если ($ w >= $ h ) {
1191    если ($ максширина && $ w >= $ максширина ) {
1192      $ отношение = $ максимальнаяширина / $ w ;
1193        } elseif ($ maxheight && $ h > $ maxheight ) {
1194      $ отношение = $ максимальная высота / $ h ;
1195        }
1196    } elseif ($ maxheight && $ h >= $ maxheight ) {
1197    $ отношение = $ максимальная высота / $ h ;
1198    } elseif ($ maxwidth && $ w > $ maxwidth ) {
1199    $ отношение = $ максимальнаяширина / $ w ;
1200    }
1201  если ($ отношение ) {
1202        $ w = пол ($ отношение * $ w );
1203        $ h = пол ( соотношение $ * $ h );
1204    }
1205
1206    //подготовить URL-адреса
1207    $ url = ml ($ IMG , [ 'cache' => $ INPUT -> str ( 'cache' ), 'rev'
=> $ REV ], true , '&' );
1208    $ src = ml ($ IMG , [ 'cache' => $ INPUT -> str ( 'cache' ), 'rev'
=> $ REV , 'w' => $ w , 'h' => $ h ], true , '&' );
1209
1210    //подготовить атрибуты
1211    $ alt = tpl_img_getTag ( 'Простой.Заголовок' );
1212  если ( is_null ($ params )) {
1213    $ p = [];
1214    } еще {
1215        $ p = $ параметры ;
1216    }
1217  если ($ w ) $ p [ 'ширина' ] = $ w ;
1218  если ($ h ) $ p [ 'высота' ] = $ h ;
```

```
1219     $ p [ 'class' ] = 'img_detail' ;
1220   если ($ альт ) {
1221         $ p [ 'alt' ] = $ alt ;
1222         $ p [ 'title' ] = $ alt ;
1223     } еще {
1224         $ p [ 'alt' ] = '' ;
1225     }
1226     $ p [ 'источник' ] = $ источник ;
1227 
1228     $ data = [ 'url' => ($ link ? $ url : null ), 'params' => $ p ];
1229     return  Event :: createAndTrigger ( 'TPL_IMG_DISPLAY' , $ data ,
'_tpl_img_action' , true );
1230}
1231 
1232/**
1233* Действие по умолчанию для  TPL_IMG_DISPLAY
1234*
1235* @param   массив $ данные
1236* @return  bool
1237*/
1238функция  _tpl_img_action ($ data )
1239{
1240   глобальный $ lang ;
1241     $ p = buildAttributes ($ data [ 'params' ]);
1242 
1243   если ($ data [ 'url' ]) echo  ' <a href="' . hsc ($ data [ 'url' ]) .
'" title="' . $ lang [ 'mediaview' ] . '"> ' ;
1244     echo  '<img ' . $ p . '/>' ;
1245   если ($ data [ 'url' ]) echo  '</a>' ;
1246   вернуть истину ;
1247}
1248 
1249/**
1250* Эта функция вставляет небольшой  gif-файл, который на самом деле является
функцией индексатора.
1251*
1252* Должен вызываться где-то в самом конце шаблона  main.php
1253*
1254* @return  bool
1255*/
1256функция  tpl_indexerWebBug ()
1257{
1258   глобальный $ ID ;
1259 
1260   $ p = [];
1261     $ p [ 'src' ] = DOKU_BASE . ' lib / exe / taskrunner.php ?id=' .
rawurlencode ($ ID ) .
1262         '&' . время ();
1263     $ p [ 'width' ] = 2 ; //больше никаких изображений размером  1x1  пиксель,
потому что мы живем во времена блокировщиков рекламы...
1264     $ p [ 'высота' ] = 1 ;
```

```
1265    $ p [ 'alt' ] = '' ;
1266    $ att = buildAttributes ($ p );
1267    echo  "<img $ att />" ;
1268  вернуть истину ;
1269}
1270
1271/**
1272* tpl_getConf($id)
1273*
1274* используйте эту функцию для доступа к переменным конфигурации шаблона
1275*
1276* @param  string $ id имя значения для доступа
1277* @param  mixed $ notset что возвращать, если настройка недоступна
1278* @return  смешанный
1279*/
1280функция  tpl_getConf ($ id , $ notset = false )
1281{
1282  глобальная $ conf ;
1283  статический $ tpl_configloaded = false ;
1284
1285    $ tpl = $ conf [ 'шаблон' ];
1286
1287  если (!$ tpl_configloaded ) {
1288        $ tconf = tpl_loadConfig ();
1289    если ($ tconf !== false ) {
1290            foreach ($ tconf  as $ key => $ value ) {
1291        если ( isset ($ conf [ 'tpl' ][$ tpl ][$ key ])) продолжить ;
1292                $ conf [ 'tpl' ][$ tpl ][$ key ] = $ value ;
1293            }
1294            $ tpl_configloaded = true ;
1295        }
1296    }
1297
1298  вернуть $ conf [ 'tpl' ][$ tpl ][$ id ] ?? $ notset ;
1299}
1300
1301/**
1302* tpl_loadConfig()
1303*
1304* считывает все переменные конфигурации шаблона
1305* эта функция автоматически вызывается  tpl_getConf()
1306*
1307* @return  false | массив
1308*/
1309функция  tpl_loadConfig ()
1310{
1311
1312    $ file = tpl_incdir (
 ) . ' /conf/default.php ' ;1313    $ conf = [];
1314
1315  если (! file_exists ($ file )) вернуть false ;
```

```
1316
1317     // загрузить файл конфигурации по умолчанию
1318   включить ($ файл );
1319
1320   вернуть $ conf ;
1321}
1322
1323// методы языка
1324
1325/**
1326* tpl_getLang($id)
1327*
1328* используйте эту функцию для доступа к переменным языка шаблона
1329*
1330* @param  string $ id ключ языковой строки
1331* @возвращаемая строка
1332*/
1333функция  tpl_getLang ($ id )
1334{
1335   статический $ lang = [];
1336
1337   если ( количество ($ язык ) === 0 ) {
1338         global $ conf , $ config_cascade ; // определенно не вызывайте
"global $lang"
1339
1340         $ path = tpl_incdir () . 'lang/' ;
1341
1342         $ lang = [];
1343
1344         // не включайте один раз
1345         @include ( $ path.'en / lang.php ' ) ;
1346           foreach ($ config_cascade [ 'lang' ][ 'template' ] как $
config_file ) {
1347       если ( file_exists ($ config_file . $ conf [ 'template' ] . '/ en /
lang.php ' )) {
1348                 include ($ config_file .$ conf [ 'template' ]. '/ en /
lang.php ' );
1349             }
1350         }
1351
1352         if ($ conf [ 'lang' ] != 'en' ) {
1353             @include ($ path .$ conf [ 'lang' ] . '/ lang.php ' );
1354             foreach ($ config_cascade [ 'lang' ][ 'template' ] как $
config_file ) {
1355         если ( file_exists ($ config_file . $ conf [ 'template' ] . '/' .
$ conf [ 'lang' ] . '/ lang.php ' )) {
1356                     include ($ config_file .$ conf [ 'template' ]. '/'
.$ conf [ 'lang' ]. '/ lang.php ' );
1357                 }
1358             }
1359         }
```

```
1360     }
1361   вернуть $ lang [$ id ] ?? '' ;
1362}
1363
1364/**
1365* Извлечь файл, зависящий от языка, и передать его в  xhtml-рендерер для
отображения
1366* эквивалент шаблона  p_locale_xhtml()
1367*
1368* @param   string $ id  идентификатор страницы вики, зависящей от языка
1369* @return    string       анализирует содержимое страницы вики в формате  xhtml
1370*/
1371функция   tpl_locale_xhtml ($ id )
1372{
1373   вернуть   p_cached_output ( tpl_localeFN ($ id ));
1374}
1375
1376/**
1377* Добавляет соответствующий путь к имени файла, зависящему от языка
1378*
1379* @param   string $ id  идентификатор локализованного текста
1380* @return   string  вики-текст
1381*/
1382функция   tpl_localeFN ($ id )
1383{
1384     $ path = tpl_incdir () . 'lang/' ;
1385   глобальная $ conf ;
1386     $ file = DOKU_CONF.'template_lang /'.$ conf [ 'template' ] . '/' .$
conf [ ' lang' ]. '/' .$ id . '.txt' ;
1387   если (! file_exists ($ file )) {
1388         $ file = $ path .$ conf [ 'lang' ]. '/' .$ id . '.txt' ;
1389     если (! file_exists ($ file )) {
1390             //возвращаемся к английскому
1391             $ file = $ path . 'en/' . $ id . '.txt' ;
1392         }
1393     }
1394   вернуть $ файл ;
1395}
1396
1397/**
1398* выводит «основной контент» во всплывающем окне медиаменеджера
1399*
1400* В зависимости от действий пользователя это может быть список
1401* файлы в пространстве имен, диалоговое окно редактирования метаданных или
1402* сообщение о ссылках на страницы
1403*
1404* Разрешено только в  mediamanager.php
1405*
1406* @triggers  МЕДИАМЕНЕДЖЕР_КОНТЕНТ_ВЫВОД
1407* @param   bool $ fromajax - установить  true  при вызове этой функции через
ajax
```

```
1408* @param  string $ сортировка
1409*
1410* @автор Андреас Гор <andi@splitbrain.org>
1411*/
1412функция  tpl_mediaContent ($ fromajax = false , $ sort = 'natural' )
1413{
1414  глобальный $ IMG ;
1415  глобальная $ AUTH ;
1416  глобальный $ INUSE ;
1417  глобальный $ NS ;
1418  глобальный $ JUMPTO ;
1419    /** @var  Вход $INPUT */
1420  глобальный $ ВХОД ;
1421
1422    $ do = $ INPUT -> extract ( 'do' )-> str ( 'do' );
1423  если ( in_array ( $ do , [ 'сохранить' , 'отмена' ])) $ do = '' ;
1424
1425  если (!$ сделать ) {
1426    если ($ INPUT -> bool ( 'редактировать' )) {
1427            $ do = 'метаформа' ;
1428        } elseif ( is_array ($ INUSE )) {
1429            $ do = 'filesinuse' ;
1430        } еще {
1431            $ do = 'список_файлов' ;
1432        }
1433    }
1434
1435    // выводим панель содержимого, обернутую в событие.
1436  если (!$ fromajax ) echo  '<div id="media__content">' ;
1437  $ данные = [ 'делать' => $ делать ];
1438    $ evt = новое событие ( 'MEDIAMANAGER_CONTENT_OUTPUT' , $ data );
1439  если ($ evt -> advice_before ()) {
1440        $ do = $ data [ 'do' ];
1441    если ($ do == 'filesinuse' ) {
1442            media_filesinuse ($ INUSE , $ IMG );
1443        } elseif ($ do == 'список файлов' ) {
1444            media_filelist ($ NS , $ AUTH , $ JUMPTO , false , $ sort );
1445        } elseif ($ do == 'searchlist' ) {
1446            media_searchlist ($ INPUT -> str ( 'q' ), $ NS , $ AUTH );
1447        } еще {
1448            msg ( 'Неизвестное действие' . hsc ($ do ), -1 );
1449        }
1450    }
1451    $ evt -> advice_after ();
1452  снято ($ evt );
1453  если (!$ fromajax ) echo  '</div>' ;
1454}
1455
1456/**
1457* Печатает центральный столбец в полноэкранном медиа-менеджере
1458* В зависимости от открытой вкладки это может быть список
```

```
1459* файлы в пространстве имен, форма загрузки или форма поиска
1460*
1461* @author Катя Арзамасцева <pshns@ukr.net>
1462*/
1463функция  tpl_mediaFileList ()
1464{
1465  глобальная $ AUTH ;
1466  глобальный $ NS ;
1467  глобальный $ JUMPTO ;
1468  глобальный $ lang ;
1469    /** @var  Вход $INPUT */
1470  глобальный $ ВХОД ;
1471
1472    $ open_tab = $ INPUT -> str ( 'tab_files' );
1473  если (!$ open_tab || ! in_array ($ open_tab , [ 'файлы' , 'загрузка' ,
'поиск' ])) $ open_tab = 'файлы' ;
1474  если ($ INPUT -> str ( 'mediado' ) == 'update' ) $ open_tab = 'upload'
;
1475
1476    echo  '<h2 class="a11y">' . $ lang [ 'mediaselect' ] . '</h2>' . NL
;
1477
1478    media_tabs_files ($ opened_tab );
1479
1480    echo  '<div class="panelHeader">' . NL ;
1481  эхо '<h3>' ;
1482    $ tabTitle = $ NS ?: '[' . $ lang [ 'mediaroot' ] . ']' ;
1483    printf ($ lang [ 'media_' . $ open_tab ], '<strong>' . hsc ($
tabTitle ) . '</strong>' );
1484  эхо '</h3>' . NL ;
1485  если ($ open_tab === 'поиск' || $ open_tab === 'файлы' ) {
1486    параметры_файлов_вкладки_медиа ();
1487    }
1488  эхо '</div>' . NL ;
1489
1490    echo  '<div class="panelContent">' . NL ;
1491  если ($ open_tab == 'файлы' ) {
1492        media_tab_files ($ NS , $ AUTH , $ JUMPTO );
1493    } elseif ($ opened_tab == 'загрузить' ) {
1494        media_tab_upload ($ NS , $ AUTH , $ JUMPTO );
1495    } elseif ($ open_tab == 'поиск' ) {
1496        media_tab_search ($ NS , $ AUTH );
1497    }
1498  эхо '</div>' . NL ;
1499}
1500
1501/**
1502* Печатает третий столбец в полноэкранном медиа-менеджере
1503* В зависимости от открытой вкладки это могут быть сведения о
1504* выбранный файл, диалоговое окно редактирования метаданных или
1505* список ревизий файлов
```

```
1506*
1507* @param  string $ изображение
1508* @param  boolean $ rev
1509*
1510* @author Катя Арзамасцева <pshns@ukr.net>
1511*/
1512функция  tpl_mediaFileDetails ($ image , $ rev )
1513{
1514  глобальный $ conf , $ DEL , $ lang ;
1515    /** @var  Вход $INPUT */
1516  глобальный $ ВХОД ;
1517
1518  $ удалено = (
1519        ! file_exists ( mediaFN ($ image )) &&
1520        file_exists ( mediaMetaFN ($ image , '.changes' )) &&
1521        $ conf [ 'mediarevisions' ]
1522    );
1523  если (!$ image || (! file_exists ( mediaFN ($ image )) && !$ removed )
|| $ DEL ) return ;
1524  если ($ rev && ! file_exists ( mediaFN ( $ image , $ rev ))) $ rev =
false ;
1525    $ ns = getNS ($ image );
1526    $ do = $ INPUT -> str ( 'mediado' );
1527
1528    $ open_tab = $ INPUT -> str ( 'tab_details' );
1529
1530    $ tab_array = [ 'view' ];
1531    [, $ mime ] = mimetype ($ image );
1532  если ($ mime == 'изображение / jpeg ' ) {
1533        $ tab_array [] = 'редактировать' ;
1534    }
1535  если ($ conf [ 'mediarevisions' ]) {
1536        $ tab_array [] = 'история' ;
1537    }
1538
1539  если (!$ open_tab || ! in_array ($ open_tab , $ tab_array )) $ open_tab
= 'view' ;
1540  если ($ INPUT -> bool ( 'редактировать' )) $ open_tab = 'редактировать' ;
1541  если ($ do == 'restore' ) $ open_tab = 'view' ;
1542
1543    media_tabs_details ($ image , $ opened_tab );
1544
1545    echo  '<div class="panelHeader"><h3>' ;
1546    [$ ext ] = mimetype ($ image , false );
1547    $ class = preg_replace ( '/[^_\-a-z0-9]+/i' , '_' , $ ext );
1548    $ class = 'выберите медиафайл mf_' . $ class ;
1549
1550  $ атрибуты = $ rev ? [ 'rev' => $ rev ] : [];
1551    $ tabTitle = sprintf (
1552        '<strong><a href="%s" class="%s" title="%s">%s</a></strong>' ,
1553    мл ($ изображение , $ атрибуты ),
```

```
1554    $ класс ,
1555        $ lang [ 'mediaview' ],
1556    $ изображение
1557    );
1558  если ($ open_tab === 'view' && $ rev ) {
1559        printf ($ lang [ 'media_viewold' ], $ tabTitle , dformat ($ rev
));
1560    } еще {
1561        printf ($ lang [ 'media_' .$ opened_tab ],$ tabTitle );
1562    }
1563
1564    echo  '</h3></div>' . NL ;
1565
1566    echo  '<div class="panelContent">' . NL ;
1567
1568  если ($ open_tab == 'view' ) {
1569        media_tab_view ($ image , $ ns , null , $ rev );
1570    } elseif ($ opening_tab == 'edit' && !$ removed ) {
1571        media_tab_edit ($ image , $ ns );
1572    } elseif ($ opened_tab == 'history' && $ conf [ 'mediarevisions' ])
{
1573        media_tab_history ($ image , $ ns );
1574    }
1575
1576  эхо '</div>' . NL ;
1577}
1578
1579/**
1580* выводит дерево пространства имен во всплывающем окне медиаменеджера
1581*
1582* Разрешено только в  mediamanager.php
1583*
1584* @автор Андреас Гор  <andi@splitbrain.org>
1585*/
1586функция  tpl_mediaTree ()
1587{
1588  глобальный $ NS ;
1589  эхо '<div id="media__tree">' ;
1590    media_nstree ($ NS );
1591  эхо '</div>' ;
1592}
1593
1594/**
1595* Распечатать выпадающее меню со всеми действиями  DokuWiki
1596*
1597* Примечание: здесь не будут использоваться красивые  URL-адреса.
1598*
1599* @param  string $  пусто пустая метка параметра
1600* @param  string $  кнопка подписи кнопки отправки
1601*
1602* @автор Андреас Гор  <andi@splitbrain.org>
```

```
1603* @deprecated 2017-09-01 см. devel:menus
1604*/
1605функция  tpl_actiondropdown ($ empty = '' , $ button = '>' )
1606{
1607    dbg_deprecated ( 'см. devel:menus' );
1608    $ menu = new  MobileMenu ();
1609    echo $ menu -> getDropdown ($ empty , $ button );
1610}
1611
1612/**
1613* Распечатать информационную строку об использованной лицензии
1614*
1615* @param  string $ img распечатать изображение? (|кнопка|значок)
1616* @param  bool $ imgonly пропустить текстовое описание?
1617* @param  bool $ return, если true, не печатать, а возвращать HTML
1618* @param  bool $ обернуть в div с class="license"?
1619* @возвращаемая строка
1620*
1621* @автор Андреас Гор <andi@splitbrain.org>
1622*/
1623функция  tpl_license ($ img = 'badge' , $ imgonly = false , $ return =
false , $ wrap = true )
1624{
1625  глобальная лицензия $ ;
1626  глобальная $ conf ;
1627  глобальный $ lang ;
1628  если (!$ conf [ 'license' ]) вернуть '' ;
1629  если (! is_array ($ license [$ conf [ 'license' ]])) вернуть '' ;
1630    $ lic = $ license [$ conf [ 'license' ]];
1631    $ target = ($ conf [ 'target' ][ 'extern' ]) ? ' target="' . $ conf
[ 'target' ][ 'extern' ] . '"' : '' ;
1632
1633    $ out = '' ;
1634  если ($ wrap ) $ out .= '<div class="license">' ;
1635  если ($ img ) {
1636        $ src = license_img ($ img );
1637    если ($ ист ) {
1638            $ out .= '<a href="' . $ lic [ 'url' ] . '" rel="лицензия"' .
$ target ;
1639            $ out .= '><img src="' . DOKU_BASE . $ src . '" alt="' . $
lic [ 'name' ] . '" /></a>' ;
1640      если (!$ imgonly ) $ out .= ' ' ;
1641        }
1642    }
1643  если (!$ imgonly ) {
1644        $ out .= $ lang [ 'license' ] . ' ' ;
1645        $ out .= '<bdi><a href="' . $ lic [ 'url' ] . '" rel="license"
class="urlextern"' . $ target ;
1646        $ out .= '>' . $ lic [ 'имя' ] . '</a></bdi>' ;
1647    }
1648  если ($ wrap ) $ out .= '</div>' ;
```

```
1649
1650  если ($ return ) вернуть $ out ;
1651  вывести $ ;
1652   возвращаться '' ;
1653}
1654
1655/**
1656* Включает визуализированный HTML-код указанной страницы
1657*
1658* Эта функция полезна для заполнения боковых панелей или подобных функций в
1659* шаблон
1660*
1661* @param  string $ pageid Имя страницы, которую вы хотите включить
1662* @param  bool $ print Следует ли печатать содержимое или только возвращать
его
1663* @param  bool $ propagate Искать также и в более высоких пространствах имен?
1664* @param  bool $ useacl Включать страницу только в том случае, если списки
контроля доступа проверены?
1665* @return  bool | null | строка
1666*/
1667функция  tpl_include_page ($ pageid , $ print = true , $ propagate =
false , $ useacl = true )
1668{
1669  если ($ распространять ) {
1670        $ pageid = page_findnearest ($ pageid , $ useacl );
1671    } elseif ($ useacl && auth_quickaclcheck ($ pageid ) == AUTH_NONE )
{
1672    вернуть ложь ;
1673    }
1674  если (!$ pageid ) вернуть false ;
1675
1676  глобальный $ TOC ;
1677    $ oldtoc = $ TOC ;
1678    $ html = p_wiki_xhtml ($ pageid , '' , false );
1679    $ TOC = $ oldtoc ;
1680
1681  если ($ print ) echo $ html ;
1682  вернуть $ html ;
1683}
1684
1685/**
1686* Отобразить форму подписки
1687*
1688* @автор Адриан Лэнг <lang@cosmocode.de>
1689* @устаревший 2020-07-23
1690*/
1691функция  tpl_subscribe ()
1692{
1693    dbg_deprecated ( Подписаться :: класс . '::show()' );
1694    ( новая Подписка ())-> показать ();
1695}
```

```
1696
1697/**
1698* Пытается отправить уже созданный контент прямо в браузер
1699*
1700* Оборачивает ob_flush() и flush()
1701*
1702* @автор Андреас Гор <andi@splitbrain.org>
1703*/
1704функция  tpl_flush ()
1705{
1706  если ( ob_get_level () > 0 ) ob_flush ();
1707  румянец ();
1708}
1709
1710/**
1711* Пытается найти файл ресурсов в указанных местах.
1712*
1713* Если указанное местоположение начинается с двоеточия, предполагается, что это
медиа
1714* файл, в противном случае предполагается, что он относится к текущему шаблону
1715*
1716* @параметр  строка []$ поиск мест для просмотра
1717*  @param  bool $ abs , если использовать абсолютный URL
1718*  @param  массив     &$imginfo заполнен с помощью getimagesize()
1719*  @param  bool $ fallback использовать резервное изображение, если цель не
найдена, или вернуть «false», если она потенциальная
1720* требуется ложный результат
1721* @возвращаемая  строка
1722*
1723* @автор Андреас Гор <andi@splitbrain.org>
1724*/
1725функция  tpl_getMediaFile ($ search , $ abs = false , &$ imginfo = null
, $ fallback = true )
1726{
1727    $ img = '' ;
1728  $ файл = '' ;
1729    $ ismedia = false ;
1730    // перебираем кандидатов, пока не будет найдено совпадение:
1731    foreach ($ поиск как $ img ) {
1732    если ( str_starts_with ($ img , ':' )) {
1733            $ file = mediaFN ($ img );
1734            $ ismedia = true ;
1735        } еще {
1736            $ file = tpl_incdir () . $ img ;
1737            $ ismedia = false ;
1738        }
1739
1740    если ( file_exists ($ file )) прерывание ;
1741    }
1742
1743    // управлять несуществующей целью
```

```
1744  если (! file_exists ($ file )) {
1745         // дать результат для резервного изображения
1746    если ($ откат ) {
1747             $ file = DOKU_INC . ' lib / images / blank.gif ' ;
1748             // остановить процесс, если требуется ложный результат (если
$fallback равен false)
1749         } еще {
1750      вернуть ложь ;
1751         }
1752    }
1753
1754    // извлечь данные изображения, если требуется
1755  если (! is_null ($ imginfo )) {
1756         $ imginfo = getimagesize ($ file );
1757    }
1758
1759    // создать URL
1760  если ($ ismedia ) {
1761         $ url = ml ($ img , '' , true , '' , $ abs );
1762    } еще {
1763         $ url = tpl_basedir (). $ изображение ;
1764    если ($ abs ) $ url = DOKU_URL.substr ( $ url , strlen ( DOKU_REL ))
 ;1765    }
1766
1767  вернуть $ url ;
1768}
1769
1770/**
1771* PHP включает файл
1772*
1773* либо из каталога conf, если он существует, в противном случае используйте
1774* файл в корневом каталоге шаблона.
1775*
1776* Функция учитывает настройки каскада конфигураций и ищет заданные
1777* файл рядом с «основными» файлами конфигурации, в порядке защищенный,
локальный,
1778* по умолчанию.
1779*
1780* Примечание: здесь не выполняется экранирование или проверка на работоспособность.
Никогда не передавайте пользовательский ввод
1781* к этой функции!
1782*
1783* @param  string $ файл
1784*
1785* @автор Андреас Гор <andi@splitbrain.org>
1786* @автор Аника Хенке <anika@selfthinker.org>
1787*/
1788функция  tpl_includeFile ($ файл )
1789{
1790  глобальный $ config_cascade ;
1791    foreach ([ 'protected' , 'local' , 'default' ] как $ config_group )
```

```
{
1792    если ( пусто ($ config_cascade [ 'main' ][$ config_group ])) продолжить
;
1793        foreach ($ config_cascade [ 'main' ][$ config_group ] как $
conf_file ) {
1794            $ dir = имя_каталога ($ conf_file );
1795      если ( file_exists ( " $ dir / $ file " )) {
1796                include ( " $ dir / $ file " );
1797        возвращаться ;
1798            }
1799        }
1800    }
1801
1802    // все еще здесь? попробуйте шаблон dir
1803    $ file = tpl_incdir () . $ file ;
1804  если ( file_exists ($ file )) {
1805    включить ($ файл );
1806    }
1807}
1808
1809/**
1810* Возвращает тег <link> для различных типов иконок (favicon|mobile|generic)
1811*
1812* @param  array $ types - список типов иконок для отображения
(favicon|mobile|generic)
1813* @возвращаемая строка
1814*
1815* @автор Аника Хенке <anika@selfthinker.org>
1816*/
1817функция  tpl_favicon ($ types = [ 'favicon' ])
1818{
1819
1820    $ return = '' ;
1821
1822    foreach ($ типы  как $ тип ) {
1823    переключатель ($ тип ) {
1824      случай 'favicon' :
1825                $ look = [ ':wiki:favicon.ico' , ':favicon.ico' , '
изображения / favicon.ico ' ];
1826                $ return .= '<link rel="значок ярлыка" href="' .
tpl_getMediaFile ($ look ) . '" />' . NL ;
1827        перерыв ;
1828      случай «мобильный» :
1829                $ look = [ ':wiki:apple-touch-icon.png' , ':apple-touch-
icon.png' , 'изображения / apple-touch-icon.png ' ];
1830                $ return .= '<link rel="apple-touch-icon" href="' .
tpl_getMediaFile ($ look ) . '" />' . NL ;
1831        перерыв ;
1832      случай «общий» :
1833                // идеальное решение, которое пока не работает ни в одном браузере
1834                $ look = [ ':wiki:favicon.svg' , ':favicon.svg' , '
```

```
изображения / favicon.svg ' ];
1835                $ return .= '<link rel="icon" href="' . tpl_getMediaFile
($ look ) . '" type=" image / svg +xml" />' . NL ;
1836        перерыв ;
1837         }
1838     }
1839
1840  возврат $ возврат ;
1841}
1842
1843/**
1844* Печать полноэкранного медиа-менеджера
1845*
1846* @author Катя Арзамасцева <pshns@ukr.net>
1847*/
1848функция  tpl_media ()
1849{
1850  глобальные $ NS , $ IMG , $ JUMPTO , $ REV , $ lang , $ fullscreen , $
INPUT ;
1851  $ полноэкранный = правда ;
1852    require_once  DOKU_INC . ' lib / exe / mediamanager.php ' ;
1853
1854    $ rev = '' ;
1855    $ image = cleanID ($ INPUT -> str ( 'image' ));
1856  если ( isset ($ IMG )) $ image = $ IMG ;
1857  если ( isset ($ JUMPTO )) $ image = $ JUMPTO ;
1858  если ( isset ($ REV ) && !$ JUMPTO ) $ rev = $ REV ;
1859
1860    echo  '<div id="mediamanager__page">' . NL ;
1861  эхо '<h1>' . $ язык [ 'btn_media' ] . '</h1>' . НЛ ;
1862    html_msgarea ();
1863
1864    echo  '<div class="panel namespaces">' . NL ;
1865    echo  '<h2>' . $ lang [ 'пространства имен' ] . '</h2>' . NL ;
1866    echo  '<div class="panelHeader">' ;
1867    echo $ lang [ 'media_namespaces' ];
1868  эхо '</div>' . NL ;
1869
1870    echo  '<div class="panelContent" id="media__tree">' . NL ;
1871    media_nstree ($ NS );
1872  эхо '</div>' . NL ;
1873  эхо '</div>' . NL ;
1874
1875    echo  '<div class="panel filelist">' . НЛ ;
1876    tpl_mediaFileList ();
1877  эхо '</div>' . NL ;
1878
1879    echo  '<div class="panel file">' . NL ;
1880    echo  '<h2 class="a11y">' . $ lang [ 'media_file' ] . '</h2>' . NL ;
1881    tpl_mediaFileDetails ($ image , $ rev );
1882  эхо '</div>' . NL ;
```

```
1883
1884  эхо '</div>' . NL ;
1885}
1886
1887/**
1888* Возвращаем полезные классы макета
1889*
1890* @возвращаемая строка
1891*
1892* @автор Аника Хенке <anika@selfthinker.org>
1893*/
1894функция  tpl_classes ()
1895{
1896  глобальные $ ACT , $ conf , $ ID , $ INFO ;
1897    /** @var  Вход $INPUT */
1898  глобальный $ ВХОД ;
1899
1900  $ классы = [
1901        'dokuwiki' ,
1902        'mode_' . $ ACT ,
1903        'tpl_' . $ conf [ 'шаблон' ],
1904        $ INPUT -> сервер -> bool ( 'REMOTE_USER' ) ? 'loggedIn' : '' ,
1905        ( isset ($ INFO [ 'существует' ]) && $ INFO [ 'существует' ]) ? ''
: 'notFound' ,
1906        ($ ID == $ conf [ 'start' ]) ? 'home' : ''
1907    ];
1908    return  implode ( ' ' , $ classes );
1909}
1910
1911/**
1912* Создать событие для меню инструментов
1913*
1914* @param  string $ toolsname имя меню
1915* @param  массив $ элементы
1916* @param  string $ view например 'main', 'detail', ...
1917*
1918* @автор Аника Хенке <anika@selfthinker.org>
1919* @deprecated 2017-09-01 см. devel:menus
1920*/
1921функция  tpl_toolsevent ($ toolsname , $ items , $ view = 'main' )
1922{
1923    dbg_deprecated ( 'см. devel:menus' );
1924    $ data = [ 'view' => $ view , 'items' => $ items ];
1925
1926    $ hook = 'TEMPLATE_' . strtoupper ($ toolsname ) . '_DISPLAY' ;
1927    $ evt = новое событие ($ hook , $ data );
1928  если ($ evt -> advice_before ()) {
1929        foreach ($ evt -> data [ 'items' ] as $ html ) echo $ html ;
1930    }
1931    $ evt -> advice_after ();
1932}
```

```
1933
```

From:
https://wwoss.ru/ - **worldwide open-source software**

Permanent link:
https://wwoss.ru/doku.php?id=wiki:xref:dokuwiki:inc:template.php

Last update: **2024/08/26 01:24**