

## main variables

**\$site**, **\$pages**, **\$page**  
available in all templates and snippets

### \$site

contains all site relevant information

**\$site->yourvar()**

the site object can contain any number of custom variables, defined in content/site.txt

**\$site->modified()**

the timestamp of the last content update

**\$site->pages()**

equals the global \$pages variable

### \$site->uri()

the uri object provides full access to the current uri in the browser address field

**\$site->uri()->path()**

get the full path as string

**\$site->uri()->path(\$n)**

get a section of the path by number

**\$site->uri()->path()->first()**

get the first section of the path

**\$site->uri()->path()->last()**

get the last section of the path

**\$site->uri()->params()**

get the entire params object

**\$site->uri()->params(\$key)**

get a single param by key

**\$site->uri()->query()**

get the entire query object

**\$site->uri()->query(\$key)**

get a query variable by key

### \$pages

first level of pages.

starting point to traverse the pages tree.

**\$pages->find(\$uri, [\$anotherUri])**

finds a subpage by uri.

returns either a single page object, a set of pages if you pass more than one uris or false if nothing has been found. you can go down the entire tree by using a uri path: "projects/project-1/subpage" etc.

**\$pages->active()**

equals \$page. returns the current page obj.

**\$pages->findOpen()**

returns the active page obj in the first level

**\$pages->findBy(\$key, \$value)**

finds a page by a defined key and value

**\$pages->findByUID(\$uid)**

finds a page by its uid.

uid: foldername without number

**\$pages->findByDirname(\$dirname)**

finds a page by its full dirname incl. number

**\$pages->findByTitle(\$title)**

finds a page by its title

**\$pages->filterBy(\$key, \$value, \$split=false)**

filters a set of pages by a key and a value.

if you pass a splitting character as third arg. it will try to split a single value. ie. a comma

seperated list of tags and search within its elements. returns the filtered set of pages.

**\$pages->first()**

returns the first page obj in a set of pages

**\$pages->last()**

returns the last page obj in a set of pages

**\$pages->count()**

counts the num. of page objs in a pages set

**\$pages->visible()**

returns all visible pages from a pages set

**\$pages->countVisible()**

counts visible pages in a set

**\$pages->invisible()**

returns all invisible pages from a pages set

**\$pages->countInvisible()**

counts invisible pages in a set

**\$pages->without(\$uid)**

returns pages without a page obj defined by its uid.

**\$pages->not(\$uid)**

equals \$pages->without(\$uid)

**\$pages->slice(\$offset, \$limit)**

slices a set of pages and returns the remaining page objs.

**\$pages->limit(\$limit)**

returns a limited number of page objs.

**\$pages->offset(\$offset)**

returns pages in set, starting from an offset

**\$pages->flip()**

flips the current order of pages in a set

**\$pages->sortBy(\$sort='title', \$dir='asc')**

sorts pages in a set by field

**\$pages->shuffle()**

shuffles the order in a set of pages

### \$pages->pagination()

add a pagination object to a set of pages, to get a set of pagination methods.

**\$list = \$pages->paginate(\$numPages)**

adds the pagination object to a set of pages

**\$list->pagination()->page()**

returns the currently active page number

**\$list->pagination()->hasPages()**

checks if the set is large enough to split it in multiple pages.

**\$list->pagination()->countPages()**

returns the number of available pages

**\$list->pagination()->countItems()**

returns the number of items/page objs in a set of pages

**\$list->pagination()->pageURL(\$page)**

returns the url for a given page

**\$list->pagination()->firstPage()**

returns the number of the first page

**\$list->pagination()->isFirstPage()**

checks if we are on the first page

**\$list->pagination()->firstPageURL()**

returns the url for the first page

**\$list->pagination()->lastPage()**

returns the number of the last page

**\$list->pagination()->isLastPage()**

checks if we are on the last page

**\$list->pagination()->lastPageURL()**

returns the url for the last page

**\$list->pagination()->prevPage()**

returns the number of the previous page

**\$list->pagination()->hasPrevPage()**

checks if there is a previous page

**\$list->pagination()->prevPageURL()**

returns the url for the previous page

**\$list->pagination()->nextPage()**

returns the number of the next page

**\$list->pagination()->hasNextPage()**

checks if there is a next page

**\$list->pagination()->nextPageURL()**

returns the url for the next page

### \$page

the currently active page object

**\$page->title()**

returns the title of the current page. if there's no title defined in the content text file, the uid will be returned.

**\$page->yourvar()**

the page object can contain any number of custom variables, defined in the content text file. you can get them by using the field names as method names: \$page->myfield()

**\$page->parent()**

returns the parent page object if available. returns false for pages on the first level without a parent.

**\$page->children()**

returns all subpages in a set (see \$pages)

**\$page->hasChildren()**

checks if there are subpages for this page

**\$page->siblings()**

returns all siblings in a set (see \$pages)

**\$page->template()**  
returns the name of the used template

**\$page->next()**  
returns the next page object

**\$page->hasNext()**  
checks if there's a next page object

**\$page->prev()**  
returns the previous page object

**\$page->hasPrev()**  
checks if there's a previous page object

**\$page->nextVisible()**  
returns the next visible page object

**\$page->hasNextVisible()**  
checks if there's a next visible page object

**\$page->prevVisible()**  
returns the previous visible page object

**\$page->hasPrevVisible()**  
checks if there's a previous visible page

**\$page->url()**  
returns the full url of the current page

**\$page->tinyurl()**  
returns the tiny url of the current page

**\$page->date(\$format=false)**  
returns the date as timestamp if a date field is available in the content text file.

**\$page->isHomePage()**  
checks if the current page is the homepage

**\$page->isErrorPage()**  
checks if the current page is the error page

**\$page->isActive()**  
checks if the current page is active.

**\$page->isOpen()**  
checks if the current page is open. open means that itself or any of its subpages and descendants is active.

**\$page->isVisible()**  
checks if the current page is visible. only pages, which have a number prefix in front of their directory name, are marked visible.

**\$page->isChildOf(\$obj)**  
checks if the current page is a child of a passed page object

**\$page->isAncestorOf(\$obj)**  
checks if the current page is an ancestor of a passed page object

**\$page->isDescendantOf(\$obj)**  
checks if the current page is a descendant of a passed page object

**\$page->isDescendantOfActive()**  
checks if the current page is a descendant of the active page object

**\$page->files()**  
returns all files (see files) for the current page

**\$page->hasFiles()**  
checks if there a files available for this page

**\$page->images()**  
returns all image files for a page

**\$page->hasImages()**  
checks if there are image files

**\$page->videos()**  
returns all video files for a page

**\$page->hasVideos()**  
checks if there are video files

**\$page->documents()**  
returns all documents for a page

**\$page->hasDocuments()**  
checks if there are documents

**\$page->sounds()**  
returns all sound files for a page

**\$page->hasSounds()**  
checks if there are sound files

---

### files

a set of files for a certain page, stored in the same folder as the content text file.

**\$page->files()->find(\$filename)**  
returns a file from set of files by its name.

**\$page->files()->findBy(\$key, \$value)**  
finds a file in a set by key and value

**\$page->files()->filterBy(\$key, \$value)**  
filters a set of files by key and value

**\$page->files()->first()**  
returns the first file in a set

**\$page->files()->last()**  
returns the last file in a set

**\$page->files()->shuffle()**  
shuffles the order in a set of files

**\$page->files()->flip()**  
flips the order of a set of files

---

### file

a single file object

**\$file->yourvar()**  
a file object can contain any number of custom variables, defined in the meta text file. (ie. myimage.jpg.txt) you can get them by using the field names as method names:  
\$page->myfield()

**\$file->name()**  
the name of the file without extension

**\$file->filename()**  
the full filename

**\$file->extension()**  
the extension of the file

**\$file->root()**  
the full path to the file on the server.

**\$file->url()**  
the full url to a file

**\$file->modified(\$format=false)**  
the last modified timestamp

**\$file->type()**  
the file type: image, video, document, sound, video or other.

**\$file->next()**  
returns the next file

**\$file->hasNext()**  
checks if there's a next file

**\$file->prev()**  
returns the previous file

**\$file->hasPrev()**  
checks if there's a previous file

**\$file->size()**  
returns the raw file size

**\$file->niceSize()**  
returns the file size in a human readable way

**\$file->mime()**  
returns the mime type if available.

---

### image

the image object is an extended version of the file object. whenever you are working with images you can use the following additional methods.

**\$image->width()**  
returns the width of the image in pixels

**\$image->height()**  
returns the height of the image in pixels

**\$image->fit(\$maxSize, \$scale=false)**  
recalculates the image size to fit in a box, defined by maxSize. If you set scale to true, smaller images will be scaled up. this does not affect the real size of the image. only \$image->width() and \$image->height() will return recalculated values after applying this. if you are looking for a way to resize images on the fly, check out the thumbs plugin. <http://getkirby.com/downloads>

**\$image->fitWidth(\$width, \$scale=false)**  
see \$images->fit()

**\$image->fitHeight(\$height, \$scale=false)**  
see \$images->fit()

---

### cheat sheet (v1)

please go to <http://getkirby.com/docs> to get the full documentation and check out the tutorials on <http://getkirby.com/tutorials>.

<http://getkirby.com>  
<http://twitter.com/getkirby>