Attribute access

How $a.b and $a[b] are resolved - arrays by key, objects through attributes(), hasAttribute() and attribute(), eZPersistentObject fields and function attributes, dynamic keys, numeric indexes, attributes after an...

Source file: `doc/eng-us/help/template-language/attribute-access.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/template-language/attribute-access.md

`$node.parent.name` reads like a chain of array keys, but `$node` is a PHP object. The template engine resolves each step through one small protocol, and that protocol is what makes nodes, content objects, users and every other kernel object usable in templates. The code is the `TYPE_ATTRIBUTE` branch of `elementValue()` in `lib/eztemplate/classes/eztemplate.php`. The examples use node 90 of a development installation, the article "Privacy Policy".

Contents

1. Dots and brackets
2. How a step is resolved
3. Fields and function attributes
4. Dynamic keys
5. Indexes
6. Attributes after an operator
7. A missing attribute
8. Looking inside a value: attribute and dump
9. Making your own class readable
10. Related pages

Dots and brackets

`.name` and `['name']` are the same lookup. Inside brackets any expression may stand, so brackets are the form for computed keys; after a dot only a name, a number or a variable.

Template

```
{def $node = fetch( 'content', 'node', hash( 'node_id', 90 ) )}
[{$node.name}] [{$node['name']}] [{$node.object.id}] [{$node.parent.name}]
[{$node.data_map.title.content}] [{$node['data_map']['title'].data_type_string}]
```

Output:

```
[Privacy Policy] [Privacy Policy] [88] [Fit & Healthy]
[Privacy Policy] [ezstring]
```

How a step is resolved

For each step the engine takes the key (it must be a number, a string or a boolean, otherwise "Cannot use type ... for attribute lookup") and then:

The value is	The step does
an array	`array_key_exists( key )`, then `$value[key]`
an object with `attribute()` and `hasAttribute()`	`hasAttribute( key )`, then `attribute( key )`
an object without those methods	error "Cannot retrieve attribute of object(...), no attribute functions available"
anything else	no value

Public PHP properties are not read: only the attribute protocol counts.

Fields and function attributes

Kernel objects derive from `eZPersistentObject` (`kernel/classes/ezpersistentobject.php`). Their `definition()` lists **fields** (database columns) and **function attributes** (names mapped to methods), and `attributes()`, `hasAttribute()` and `attribute()` serve both:

- a field returns the member that holds the column, without a query;
- a function attribute calls its method, for example `children_count` calls a method that counts in the database, `data_map` loads the fields of the object, `url_alias` builds the URL.

Template

```
{def $node = fetch( 'content', 'node', hash( 'node_id', 90 ) )}
field:    node_id={$node.node_id} depth={$node.depth} path_string={$node.path_string}
function: children_count={$node.children_count} class_identifier={$node.class_identifier} url_ali
as={$node.url_alias}
object field: published={$node.object.published} section_id={$node.object.section_id}
object function: class_name={$node.object.class_name} main_node_id={$node.object.main_node_id}
```

Output:

```
field:    node_id=90 depth=3 path_string=/1/2/89/90/
function: children_count=0 class_identifier=ng_article url_alias=privacy-policy
object field: published=1674667522 section_id=1
object function: class_name=NG Article main_node_id=90
```

That difference matters for speed: a function attribute in a loop runs its method in every iteration. Read it once into a variable with `{def}` when you need it several times.

Dynamic keys

A variable in brackets, or after a dot, supplies the key:

Template

```
{def $node = fetch( 'content', 'node', hash( 'node_id', 90 ) )
     $field = 'node_id'
     $keys = array( 'name', 'depth', 'class_identifier' )}
[{$node[$field]}] [{$node.$field}]
{foreach $keys as $k}[{$k}={$node[$k]}]{/foreach}
```

Output:

```
[90] [90][name=Privacy Policy][depth=3][class_identifier=ng_article]
```

Indexes

Lists are read by position, with a number after the dot or any expression in brackets:

Template

```
{def $list = array( 'zero', 'one', 'two' ) $i = 2}
[{$list.0}] [{$list[1]}] [{$list[$i]}] [{$list[$i|dec]}]
```

Output:

```
[zero] [one] [two] [one]
```

Attributes after an operator

A dot or brackets may follow an operator call, so the result of a fetch or an array operator can be read directly:

Template

```
[{fetch( 'content', 'node', hash( 'node_id', 89 ) ).name}]
[{fetch( 'content', 'node', hash( 'node_id', 89 ) ).children|count}]
[{array( 'x', 'y', 'z' )|reverse.0}]
```

Output:

```
[Fit & Healthy]
[16]
[z]
```

A missing attribute

A key that does not exist logs an error and gives no value. For arrays and objects the message lists the keys that do exist (the first 25, from `attributes()` for objects), which makes it the quickest way to discover what a value offers:

Template

```
{def $map = hash( 'red', 1, 'green', 2 )}
[{$map.blue}]
```

Output:

```
[]
```

Logged by the template engine:

```
ERROR : No such attribute for array(2): blue
Choose one of following: red, green
```

Looking inside a value: attribute and dump

The `attribute` operator (`eZTemplateAttributeOperator`) walks a value and lists its keys and types, with values when the first parameter is `show`. Its parameters are `show_values` (`show` or anything else), `max_val` (levels to descend, default 2 for `attribute` and 1 for `dump`) and `format` (`html` or `text`, default from `template.ini` [AttributeOperator] DefaultFormatter, which is `html`). `dump` is the same operator with values shown by default.

Template

```
{hash( 'title', 'Hello', 'sizes', array( 'S', 'M' ) )|attribute( show, 2, text )}
```

Output:

```
title (string='Hello')
sizes (array=Array(2))
    0 (string='S')
    1 (string='M')
```

Template

```
{hash( 'title', 'Hello', 'count', 3 )|dump( show, 1, text )}
```

Output:

```
title (string='Hello')
count (integer=3)
```

On an object it lists every attribute the object offers, which is how the attribute names of a node can be read on your own installation. The first 12 lines for node 90 (the operator returns text, so the usual string and array operators can cut it):

Template

```
{def $node = fetch( 'content', 'node', hash( 'node_id', 90 ) )
     $list = $node|attribute( hide, 1, text )}
{$list|explode( "\n" )|extract_left( 12 )|implode( "\n" )}
```

Output:

```
node_id (integer)
parent_node_id (integer)
main_node_id (integer)
contentobject_id (integer)
contentobject_version (integer)
contentobject_is_published (integer)
depth (integer)
sort_field (integer)
sort_order (integer)
priority (integer)
modified_subnode (integer)
path_string (string)
```

Remove the operator again before the template goes live: it calls every function attribute of the object.

Making your own class readable

Any PHP object becomes readable in templates when it has the three methods. A class that extends `eZPersistentObject` gets them from its definition. A plain class implements them itself:

PHP

```php
class expExampleBox
{
    public function attributes()
    {
        return array( 'title', 'items' );
    }

    public function hasAttribute( $name )
    {
        return in_array( $name, $this->attributes() );
    }

    public function attribute( $name )
    {
        switch ( $name )
        {
            case 'title': return $this->title;
            case 'items': return $this->loadItems();
        }
        return null;
    }
}
```

This sketch was not run on a development installation; it shows the shape the engine checks for (`method_exists( $value, 'attribute' )` and `method_exists( $value, 'hasAttribute' )`).

Related pages

- Previous: Namespaces
- Next: Operators in expressions
- Attributes and dot notation

- Nodes, objects and attributes
- attribute and dump
- Free documentation: this page is free to copy, translate and improve under the GNU FDL 1.3.

> **Become an Exponential Partner**
>
> Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.
>
> Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later: https://www.gnu.org/licenses/fdl-1.3.html.