arg() is deprecated and will be removed

**D7**

<?php  
// Retrieve an array which contains the path pieces.  
$path\_args = arg();  
  
// Check if the current page is admin.  
if (arg(0) == 'admin') {  
  // This is wrong even in D7. path\_is\_admin() should be used instead.  
}  
  
// Conditionally add css or js in certain page.  
function mymodule\_page\_build(&$page) {  
  if (arg(0) == 'sth' && arg(1) == 'else') {  
    drupal\_add\_css(drupal\_get\_path('module', 'mymodule') . '/css/my.css');  
  }  
}  
  
// Load the current node.  
if (arg(0) == 'node' && is\_numeric(arg(1))) {  
  // This is wrong even in D7. menu\_get\_object() should be used instead.  
}  
?>

**D8 (procedural)**

<?php  
namespace Drupal;  
  
// Retrieve an array which contains the path pieces.  
$path\_args = explode('/', current\_path());  
  
// Check if the current page is admin.  
if (service('router.admin\_context')->isAdminRoute(\Drupal::routeMatch()->getRouteObject())) {  
}  
  
// Conditionally add css or js in certain page.  
function mymodule\_page\_attachments(array &$attachments) {  
  if (\Drupal::routeMatch()->getRouteName() == 'my.route') {  
    $attachments ['#attached']['library'][] = 'my\_module/my\_library'; // you'll need to define in mymodule.libraries.yml  
  }  
}  
  
// Load the current node.  
$node = \Drupal::routeMatch()->getParameter('node');  
if ($node) {  
}  
?>

**D8 (OOP)**

<?php  
use Symfony\Component\HttpFoundation\Request;  
use Drupal\Core\Routing\AdminContext;  
use Drupal\Core\Routing\RouteMatchInterface;  
class myClass {  
  public function \_\_construct(Request $request, AdminContext $admin\_context, RouteMatchInterface $route\_match) {  
    $this->request = $request;  
    $this->adminContext = $admin\_context;  
    $this->routeMatch = $route\_match;  
  }  
  
  public function test() {  
    // This might change in https://drupal.org/node/2239009  
    $current\_path = $this->request->attributes->get('\_system\_path');  
    // Retrieve an array which contains the path pieces.  
    $path\_args = explode('/', $current\_path);  
  
    // Check if the current page is admin.  
    if ($this->adminContext->isAdminRoute($this->routeMatch->getRouteObject())) {  
    }  
  
    // Load the current node.  
    $node = $this->routeMatch->getParameter('node');  
    if ($node) {  
    }  
  }  
}  
?>

Drupal 8: rendering a image programmatically

<?php

$file = File::load(1);

$variables = array(

'style\_name' => 'thumbnail',

'uri' => $file->getFileUri(),

);

// The image.factory service will check if our image is valid.

$image = \Drupal::service('image.factory')->get($file->getFileUri());

if ($image->isValid()) {

$variables['width'] = $image->getWidth();

$variables['height'] = $image->getHeight();

}

else {

$variables['width'] = $variables['height'] = NULL;

}

$logo\_render\_array = [

'#theme' => 'image\_style',

'#width' => $variables['width'],

'#height' => $variables['height'],

'#style\_name' => $variables['style\_name'],

'#uri' => $variables['uri'],

];  
?>

# l() and url() are removed in favor of a routing based URL generation API

Url::fromRoute($route\_name, $route\_parameters, $options)  
Url::fromUri($uri, $options)

**D7**

<?php  
// Internal path.  
$internal\_link = l(t('Book admin'), 'admin/structure/book');  
  
// External Uri.  
$external\_link = l(t('External link'), 'http://www.example.com/', array('external' => TRUE));  
?>

**D8 (procedural)**

<?php  
// Internal path (defined by a route in Drupal 8).  
use Drupal\Core\Url;  
$url = Url::fromRoute('book.admin');  
$internal\_link = \Drupal::l(t('Book admin'), $url);  
  
// External Uri.  
use Drupal\Core\Url;  
$url = Url::fromUri('http://www.example.com/');  
$external\_link = \Drupal::l(t('External link'), $url);

Direct link using url function and load from route:

\Drupal::url('entity.taxonomy\_term.add\_form', array('taxonomy\_vocabulary' => 'tags'))

**//Internal URL**

use Drupal\Core\Link;

use Drupal\Core\Url;

$options = array(

'query' => ['type' => 'article', 'status' => 1],

'fragment' => 'article-list',

'attributes' => ['class' => ['btn', 'btn-mini']],

'absolute' => TRUE,

);

$link = Link::fromTextAndUrl(t('Drupal node articles'), Url::fromUri('internal:/node', $options))->toString();

**//External URL**

use Drupal\Core\Link;

use Drupal\Core\Url;

$options = array(

'query' => ['page' => 1],

'fragment' => 'content',

'attributes' => ['class' => ['btn', 'btn-mini']],

);

$link = Link::fromTextAndUrl(t('Nginx articles'), Url::fromUri('https://www.webfoobar.com/taxonomy/term/23', $options))->toString();

?>

# Menu tree building is now a service

### Drupal 7

<?php  
// Retrieve the administration menu from the database.  
$tree = toolbar\_get\_menu\_tree();  
  
// Add attributes to the links before rendering.  
toolbar\_menu\_navigation\_links($tree);  
  
$menu = array(  
  '#heading' => t('Administration menu'),  
  'toolbar\_administration' => array(  
    '#type' => 'container',  
    '#attributes' => array(  
      'class' => array('toolbar-menu-administration'),  
    ),  
    'administration\_menu' => menu\_tree\_output($tree),  
  ),  
);  
?>

### Drupal 8 after addition of services

Example from function toolbar\_get\_rendered\_subtrees()

<?php  
  $menu\_tree = \Drupal::menuTree();  
  $parameters = new MenuTreeParameters();  
  $parameters->setRoot('system.admin')->excludeRoot()->setMaxDepth(3)->onlyEnabledLinks();  
  $tree = $menu\_tree->load(NULL, $parameters);  
  $manipulators = array(  
    array('callable' => 'menu.default\_tree\_manipulators:checkAccess'),  
    array('callable' => 'menu.default\_tree\_manipulators:generateIndexAndSort'),  
    array('callable' => 'toolbar\_menu\_navigation\_links'),  
  );  
  $tree = $menu\_tree->transform($tree, $manipulators);  
  $subtrees = array();  
  foreach ($tree as $element) {  
    /\*\* @var \Drupal\Core\Menu\MenuLinkInterface $link \*/  
    $link = $element->link;  
    if ($element->subtree) {  
      $subtree = $menu\_tree->build($element->subtree);  
      $output = drupal\_render($subtree);  
    }  
    else {  
      $output = '';  
    }  
?>

**Print Drupal 7 & 8 field value**

<?php

# Drupal 7 code.

$node->title

$node->body[$langcode][0]['value']

# Drupal 8 code.

$node->get('title')->value

$node->get('body')->getValue();

?>

**Drupal 8 Entity query:**

**Option: 1**

$storage = \Drupal::entityManager()->getStorage('node');

$nids = $storage->getQuery()

->condition('type', ‘article’)

->condition('status', 1)

->condition('field\_tags', 'drupal')

->execute();

$storage->loadMultiple($nids)

**Option: 2**

$query = \Drupal::entityQuery('node')

->condition('type', 'article')

->condition('status', 1);

$nids = $query->execute();

entity\_load\_multiple(‘node’, $nids);

**Get taxonomy terms of a vocabulary in Drupal 8**

$query = \Drupal::entityQuery('taxonomy\_term');

$query->condition('vid', "tags");

$tids = $query->execute();

$terms = \Drupal\taxonomy\Entity\Term::loadMultiple($tids);

If you want to get several vocabulary, use OR condition like that.

$query\_or->condition('vid', "tags");

$query\_or->condition('vid', "test");

$query->condition($query\_or);

Get Taxonomy name

$name = $term->toLink()->getText();

Create link to the term

$link = \Drupal::l($term->toLink()->getText(), $term->toUrl());

**Dependency Injection:**

## Static Method (Not recommended )

<?php

namespace Drupal\my\_module\Controller;

use Drupal\Core\Controller\ControllerBase;

/\*\*

\* MyModule Class defines ajax callback function.

\*/

class MyModule extends ControllerBase {

/\*\*

\* Callback function for ajax request.

\*/

public function getUserContent() {

$alias = \Drupal::service('path.alias\_manager')->getPathByAlias($\_POST['url']);

$alias = explode('/', $alias);

$my\_module\_views = views\_embed\_view('my\_module', 'default', $alias[2]);

$my\_module= drupal\_render($my\_module\_views);

return array(

'#name' => 'my\_module\_content',

'#markup' => '<div class="my\_module\_content">' . $my\_module. '</div>',

);

}

}

## Class based methods (Recommended)

<?php

namespace Drupal\my\_module\Controller;

use Drupal\Core\Controller\ControllerBase;

use Drupal\Core\Path\AliasManagerInterface;

use Symfony\Component\DependencyInjection\ContainerInterface;

/\*\*

\* MyModule Class defines ajax callback function.

\*/

class MyModule extends ControllerBase {

/\*\* @var \Drupal\Core\Path\AliasManagerInterface $aliasManager \*/

protected $aliasManager;

/\*\*

\* MyModule constructor.

\*

\* @param \Drupal\Core\Path\AliasManagerInterface $alias\_manager

\*/

public function \_\_construct(AliasManagerInterface $alias\_manager) {

$this->aliasManager = $alias\_manager;

}

/\*\*

\* {@inheritdoc}

\*/

public static function create(ContainerInterface $container) {

return new static(

$container->get('path.alias\_manager')

);

}

/\*\*

\* Callback function for ajax request.

\*/

public function getUserContent() {

$alias = $this->aliasManager->getPathByAlias($\_POST['url']);

// Your code.

}

}