Functions.php Reference Documentation

This document provides comprehensive documentation for all global utility functions available in `/includes/functions.php`.

Table of Contents

Weapon Table Formatting Functions

`formatWeaponGrade($grade)`

Formats weapon item grade to make it more readable.

Parameters:

- `$grade` (string): Raw itemGrade from weapon table

Returns: `string` - Formatted weapon grade

Example:

```php
echo formatWeaponGrade('ONLY');   // Output: "Unique"
echo formatWeaponGrade('ADVANC'); // Output: "Advanced"
```

`formatWeaponType($type)`

Formats weapon type strings to make them more readable.

Parameters:

- `$type` (string): Raw weapon type from database

Returns: `string` - Formatted weapon type

Example:

```php
echo formatWeaponType('TOHAND_SWORD'); // Output: "Two-Handed Sword"
echo formatWeaponType('SINGLE_BOW');   // Output: "Single Bow"
echo formatWeaponType('CHAINSWORD');   // Output: "Chainsword"
```

Supported Types:

- SWORD, DAGGER, TOHAND_SWORD, BOW, SPEAR, BLUNT, STAFF, STING, ARROW, GAUNTLET, CLAW, EDORYU, SINGLE_BOW, SINGLE_SPEAR, TOHAND_BLUNT, TOHAND_STAFF, KEYRINGK, CHAINSWORD

`formatWeaponMaterial($material)`

Formats weapon material strings by removing Korean text and normalizing.

Parameters:

- `$material` (string): Raw material from weapon table

Returns: `string` - Formatted material

Example:

```php
echo formatWeaponMaterial('IRON(철)');     // Output: "Iron"
echo formatWeaponMaterial('MITHRIL(미스릴)'); // Output: "Mithril"
echo formatWeaponMaterial('PLASTIC(블랙미스릴)'); // Output: "Black Mithril"
```

Armor Table Formatting Functions

`formatArmorGrade($grade)`

Formats armor item grade to make it more readable.

Parameters:

- `$grade` (string): Raw itemGrade from armor table

Returns: `string` - Formatted armor grade

Example:

```php
echo formatArmorGrade('MYTH');   // Output: "Mythical"
echo formatArmorGrade('LEGEND'); // Output: "Legendary"
```

`formatArmorType($type)`

Formats armor type strings to make them more readable.

Parameters:

- `$type` (string): Raw armor type from database

Returns: `string` - Formatted armor type

Example:

```php
echo formatArmorType('T_SHIRT'); // Output: "T-Shirt"
echo formatArmorType('HELMET');  // Output: "Helmet"
echo formatArmorType('RING_2');  // Output: "Ring (2nd)"
```

Supported Types:

- NONE, HELMET, ARMOR, T_SHIRT, CLOAK, GLOVE, BOOTS, SHIELD, AMULET, RING, BELT, RING_2, EARRING, GARDER, RON, PAIR, SENTENCE, SHOULDER, BADGE, PENDANT

`formatArmorMaterial($material)`

Formats armor material strings by removing Korean text and normalizing.

Parameters:

- `$material` (string): Raw material from armor table

Returns: `string` - Formatted material

Example:

```php
echo formatArmorMaterial('LEATHER(가죽)'); // Output: "Leather"
echo formatArmorMaterial('DRAGON_HIDE(용비늘)'); // Output: "Dragon Hide"
```

Etcitem Table Formatting Functions

`formatEtcitemGrade($grade)`

Formats etcitem grade to make it more readable.

Parameters:

- `$grade` (string): Raw itemGrade from etcitem table

Returns: `string` - Formatted item grade

Example:

```php
echo formatEtcitemGrade('HERO'); // Output: "Heroic"
echo formatEtcitemGrade('RARE'); // Output: "Rare"
```

`formatEtcitemType($type)`

Formats etcitem item_type for display.

Parameters:

- `$type` (string): Raw item_type from etcitem table

Returns: `string` - Formatted item type

Example:

php

```php
echo formatEtcitemType('FIRE_CRACKER'); // Output: "Firecracker"
echo formatEtcitemType('QUEST_ITEM');   // Output: "Quest Item"
echo formatEtcitemType('SPELL_BOOK');   // Output: "Spell Book"
```

Supported Types:

- ARROW, WAND, LIGHT, GEM, TOTEM, FIRE_CRACKER, POTION, FOOD, SCROLL, QUEST_ITEM, SPELL_BOOK, PET_ITEM, OTHER, MATERIAL, EVENT, STING, TREASURE_BOX

`formatEtcitemUseType($useType)`

Formats etcitem use_type for display.

Parameters:

- `$useType` (string): Raw use_type from etcitem table

Returns: `string` - Formatted use type

Example:

php

```php
echo formatEtcitemUseType('SPELL_LONG');    // Output: "Long Range Spell"
echo formatEtcitemUseType('INVISABLE');     // Output: "Invisibility"
echo formatEtcitemUseType('FISHING_ROD');   // Output: "Fishing Rod"
```

Supported Types (56 total):

- NONE, NORMAL, WAND1, WAND, SPELL_LONG, NTELE, IDENTIFY, RES, TELEPORT, INVISABLE, LETTER, LETTER_W, CHOICE, INSTRUMENT, SOSC, SPELL_SHORT, T_SHIRT, CLOAK, GLOVE, BOOTS, HELMET, RING, AMULET, SHIELD, GARDER, DAI, ZEL, BLANK, BTELE, SPELL_BUFF, CCARD, CCARD_W, VCARD, VCARD_W, WCARD, WCARD_W, BELT, SPELL_LONG2, EARRING, FISHING_ROD, RON, RON_2, ACCZEL, PAIR, HEALING, SHOULDER, BADGE, POTENTIAL_SCROLL, SPELLMELT, ELIXER_RON, INVENTORY_BONUS, TAM_FRUIT, RACE_TICKET, PAIR_2, MAGICDOLL, SENTENCE, SHOULDER_2, BADGE_2, PET_POTION, GARDER_2, DOMINATION_POLY, PENDANT, SHOVEL, LEV_100_POLY, SMELTING, PURIFY, CHARGED_MAP_TIME

`formatEtcitemMaterial($material)`

Formats etcitem material for display.

Parameters:

- `$material` (string): Raw material from etcitem table

Returns: `string` - Formatted material

Example:

```php
echo formatEtcitemMaterial('GEMSTONE(보석)'); // Output: "Gemstone"
echo formatEtcitemMaterial('ORIHARUKON(오리하루콘)'); // Output: "Oriharukon"
```

`formatEtcitemAttribute($attr)`

Formats etcitem attribute for display.

Parameters:

- `$attr` (string): Raw attr from etcitem table

Returns: `string` - Formatted attribute

Example:

```php
echo formatEtcitemAttribute('EARTH'); // Output: "Earth"
echo formatEtcitemAttribute('FIRE');  // Output: "Fire"
```

Supported Attributes:

- EARTH, AIR, WATER, FIRE, NONE

`formatEtcitemAlignment($alignment)`

Formats etcitem alignment for display.

Parameters:

- `$alignment` (string): Raw alignment from etcitem table

Returns: `string` - Formatted alignment

Example:

```php
echo formatEtcitemAlignment('CAOTIC');  // Output: "Chaotic"
echo formatEtcitemAlignment('LAWFUL');  // Output: "Lawful"
```

Supported Alignments:

- CAOTIC, NEUTRAL, LAWFUL, NONE

`formatEtcitemSkillType($skillType)`

Formats etcitem skill_type for display.

Parameters:

- `$skillType` (string): Raw skill_type from etcitem table

Returns: `string` - Formatted skill type

Example:

```php
echo formatEtcitemSkillType('active');  // Output: "Active"
echo formatEtcitemSkillType('passive'); // Output: "Passive"
```

Supported Skill Types:

- passive, active, none

`formatEtcitemLimitType($limitType)`

Formats etcitem limit_type for display.

Parameters:

- `$limitType` (string): Raw limit_type from etcitem table

Returns: `string` - Formatted limit type

Example:

php

```php
echo formatEtcitemLimitType('WORLD_WAR');  // Output: "World War"
echo formatEtcitemLimitType('BEGIN_ZONE'); // Output: "Begin Zone"
```

Supported Limit Types:

- WORLD_WAR, BEGIN_ZONE, NONE

Universal Formatting Functions

`formatGrade($grade)`

Formats any item grade (works for all tables).

Parameters:

- `$grade` (string): Raw itemGrade from any table

Returns: `string` - Formatted grade

Example:

php

```php
echo formatGrade('ONLY');   // Output: "Unique"
echo formatGrade('NORMAL'); // Output: "Normal"
```

Supported Grades:

- ONLY, MYTH, LEGEND, HERO, RARE, ADVANC, NORMAL

`cleanItemName($name)`

Removes color prefix codes from item names.

Parameters:

- `$name` (string): Raw item name from database (desc_en)

Returns: `string` - Cleaned item name

Example:

php

```php
echo cleanItemName('\aHDemon Sword'); // Output: "Demon Sword"
echo cleanItemName('\f4Magic Ring');  // Output: "Magic Ring"
```

Removed Prefixes:

- \aH, \aF, \f4, \aG

`formatBoolean($value)`

Formats boolean values for display.

Parameters:

- `$value` (string|int): Boolean value from database

Returns: `string` - Formatted boolean

Example:

php

```php
echo formatBoolean('true');  // Output: "Yes"
echo formatBoolean(0);       // Output: "No"
echo formatBoolean(1);       // Output: "Yes"
```

`formatMergeStatus($merge)`

Formats merge status for display.

Parameters:

- `$merge` (string): Merge status from database

Returns: `string` - Formatted merge status

Example:

php

```php
echo formatMergeStatus('true');  // Output: "Stackable"
echo formatMergeStatus('false'); // Output: "Non-stackable"
```

Image and Icon Functions

`getItemIconUrl($iconId)`

Gets the complete URL for an item icon with fallback to placeholder.

Parameters:

- `$iconId` (int): Icon ID from database

Returns: `string` - Complete URL to icon

Example:

```php
echo getItemIconUrl(1234); // Output: "https://yoursite.com/assets/img/icons/1234.png"
echo getItemIconUrl(0);    // Output: "https://yoursite.com/assets/img/placeholders/0.png"
```

`getItemSpriteUrl($spriteId)`

Gets the complete URL for an item sprite.

Parameters:

- `$spriteId` (int): Sprite ID from database

Returns: `string` - Complete URL to sprite

Example:

```php
echo getItemSpriteUrl(5678); // Output: "https://yoursite.com/assets/img/sprites/5678.png"
```

`getMonsterSpriteUrl($spriteId)`

Gets monster sprite URL with PNG fallback to GIF and ms icon.

Parameters:

- `$spriteId` (int): Monster sprite ID

Returns: `string` - Complete URL to monster sprite

Example:

```php
echo getMonsterSpriteUrl(123); // Tries PNG, then GIF, then ms123.png, then placeholder
```

Database Utility Functions

`getItemDrops($itemId, $db)`

Gets monsters that drop a specific item.

Parameters:

- `$itemId` (int): Item ID to check drops for
- `$db` (Database): Database instance

Returns: `array` - Array of drop information

Example:

```php
$db = Database::getInstance();
$drops = getItemDrops(1234, $db);
foreach ($drops as $drop) {
    echo $drop['mobname_en'] . " drops this item\n";
}
```

`hasBinData($nameId, $db)`

Checks if an item has binary data available.

Parameters:

- `$nameId` (int): Name ID to check
- `$db` (Database): Database instance

Returns: `bool` - Whether bin data exists

Example:

```php
$db = Database::getInstance();
if (hasBinData(1234, $db)) {
    echo "This item has binary data available";
}
```

`getBinItemData($nameId, $db)`

Gets binary item data for a specific name ID.

Parameters:

- `$nameId` (int): Name ID to get data for
- `$db` (Database): Database instance

Returns: `array|null` - Bin data or null if not found

Example:

```php
$db = Database::getInstance();
$binData = getBinItemData(1234, $db);
if ($binData) {
    echo "Found bin data for item";
}
```

Resistance and Attribute Functions

`formatResistanceName($resistName)`

Formats resistance field names for display.

Parameters:

- `$resistName` (string): Resistance field name

Returns: `string` - Formatted resistance name

Example:

```php
echo formatResistanceName('regist_skill');  // Output: "Skill Resistance"
echo formatResistanceName('hitup_dragon');  // Output: "Dragon Hit"
```

`getClassRestrictions($item)`

Gets formatted class restrictions for an item.

Parameters:

- `$item` (array): Item data with use_* fields

Returns: `array` - Array of class restrictions

Example:

```php
$restrictions = getClassRestrictions($weapon);
foreach ($restrictions as $class => $data) {
    echo $data['name'] . ": " . ($data['can_use'] ? 'Can' : 'Cannot') . " use\n";
}
```

Supported Classes:

- royal, knight, mage, elf, darkelf, dragonknight, illusionist, warrior, fencer, lancer

---

Item Filtering Utility Functions

`getGradeOrder()`

Gets the proper order for item grades.

Parameters: None

Returns: `array` - Grade order array

Example:

```php
$order = getGradeOrder(); // ['ONLY', 'MYTH', 'LEGEND', 'HERO', 'RARE', 'ADVANC', 'NORMAL']
```

`getGradeOrderSql($table)`

Generates SQL for ordering by grade.

Parameters:

- `$table` (string): Optional table name prefix

Returns: `string` - SQL ORDER BY clause

Example:

```php
$sql = "SELECT * FROM weapon ORDER BY " . getGradeOrderSql('weapon');
```

`getDistinctValues($db, $table, $column, $where, $params, $orderBy)`

Gets distinct values from a table column.

Parameters:

- `$db` (Database): Database instance
- `$table` (string): Table name
- `$column` (string): Column name
- `$where` (string): Optional WHERE clause
- `$params` (array): Optional parameters
- `$orderBy` (string): Optional ORDER BY clause

Returns: `array` - Array of distinct values

Example:

```php
$db = Database::getInstance();
$types = getDistinctValues($db, 'weapon', 'type');
```

Display Utility Functions

`formatStatBonus($value)`

Formats stat bonuses with + prefix for positive values.

Parameters:

- `$value` (int): Stat value

Returns: `string` - Formatted stat value

Example:

```php
echo formatStatBonus(5);   // Output: "+5"
echo formatStatBonus(-3);  // Output: "-3"
echo formatStatBonus(0);   // Output: "—"
```

`formatPercentage($value, $decimals)`

Formats percentage values.

Parameters:

- `$value` (float): Percentage value
- `$decimals` (int): Number of decimal places (default: 1)

Returns: `string` - Formatted percentage

Example:

```php
echo formatPercentage(25.5, 1);  // Output: "25.5%"
echo formatPercentage(0, 0);     // Output: "—"
```

`formatDamageRange($small, $large)`

Formats damage range display.

Parameters:

- `$small` (int): Small damage value
- `$large` (int): Large damage value

Returns: `string` - Formatted damage range

Example:

```php
echo formatDamageRange(10, 15); // Output: "10 - 15"
echo formatDamageRange(0, 0);   // Output: "—"
```

`hasStatBonuses($item, $stats)`

Checks if item has any stat bonuses.

Parameters:

- `$item` (array): Item data
- `$stats` (array): Optional array of stat fields to check

Returns: `bool` - Whether any stat bonuses exist

Example:

```php
if (hasStatBonuses($weapon)) {
    echo "This weapon has stat bonuses";
}
```

`hasResistances($item, $resistances)`

Checks if item has any resistances.

Parameters:

- `$item` (array): Item data
- `$resistances` (array): Optional array of resistance fields to check

Returns: `bool` - Whether any resistances exist

Example:

```php
if (hasResistances($armor)) {
    echo "This armor has resistances";
}
```

Helper Functions for Filtering and Sorting

`getWeaponTypes()`

Gets all possible weapon types for filtering.

Parameters: None

Returns: `array` - Array of weapon types

Example:

```php
$types = getWeaponTypes();
foreach ($types as $key => $label) {
    echo "<option value='$key'>$label</option>";
}
```

`getArmorTypes()`

Gets all possible armor types for filtering.

Parameters: None

Returns: `array` - Array of armor types

Example:

```php
$types = getArmorTypes();
// Returns: ['HELMET' => 'Helmet', 'ARMOR' => 'Armor', ...]
```

`getEtcitemTypes()`

Gets all possible etcitem types for filtering.

Parameters: None

Returns: `array` - Array of etcitem types

Example:

```php
$types = getEtcitemTypes();
// Returns: ['ARROW' => 'Arrow', 'WAND' => 'Wand', ...]
```

`getAllGrades()`

Gets all available grades for filtering.

Parameters: None

Returns: `array` - Array of grades

Example:

```php
$grades = getAllGrades();
// Returns: ['ONLY' => 'Unique', 'MYTH' => 'Mythical', ...]
```

`getAllMaterials()`

Gets all available materials for filtering.

Parameters: None

Returns: `array` - Array of materials

Example:

```php
$materials = getAllMaterials();
// Returns: ['IRON' => 'Iron', 'MITHRIL' => 'Mithril', ...]
```

Pagination Utility Functions

`getPaginationUrl($page, $params)`

Generates pagination URL with preserved parameters.

Parameters:

- `$page` (int): Page number
- `$params` (array): Additional parameters to preserve

Returns: `string` - Generated URL

Example:

```php
$url = getPaginationUrl(2, ['search' => 'sword', 'type' => 'weapon']);
// Output: "?page=2&search=sword&type=weapon"
```

`getPaginationInfo($currentPage, $perPage, $totalItems)`

Generates pagination info text.

Parameters:

- `$currentPage` (int): Current page number
- `$perPage` (int): Items per page
- `$totalItems` (int): Total number of items

Returns: `string` - Pagination info text

Example:

```php
echo getPaginationInfo(2, 20, 150); // Output: "Showing 21 to 40 of 150 items"
```

Debugging and Logging Functions

`logQuery($query, $params, $executionTime)`

Logs database queries for debugging (when DEBUG_QUERIES is enabled).

Parameters:

- `$query` (string): SQL query
- `$params` (array): Query parameters (default: [])
- `$executionTime` (float): Execution time in seconds (default: 0)

Returns: `void`

Example:

```php
logQuery("SELECT * FROM weapon WHERE type = ?", ['SWORD'], 0.002);
```

`generateCacheKey($prefix, $params)`

Generates a cache key for database results.

Parameters:

- `$prefix` (string): Cache key prefix
- `$params` (array): Parameters to include in cache key (default: [])

Returns: `string` - Generated cache key

Example:

```php
$key = generateCacheKey('weapons', ['type' => 'SWORD', 'grade' => 'HERO']);
// Output: "weapons_a1b2c3d4e5f6..." (MD5 hash)
```

Usage Notes

Best Practices

1. **Always include functions.php** in your pages:

   ```php
   require_once __DIR__ . '/../../includes/functions.php';
   ```

2. **Use table-specific functions** for better accuracy:

php

```php
// For weapons
echo formatWeaponType($weapon['type']);
echo formatWeaponGrade($weapon['itemGrade']);

// For armor
echo formatArmorType($armor['type']);
echo formatArmorGrade($armor['itemGrade']);

// For etcitems
echo formatEtcitemType($item['item_type']);
echo formatEtcitemUseType($item['use_type']);
```

3. **Use Database instance for functions that need it**:

php

```php
$db = Database::getInstance();
$drops = getItemDrops($itemId, $db);
```

4. **Check return values** for functions that might return null:

php

```php
$binData = getBinItemData($nameId, $db);
if ($binData) {
    // Process bin data
}
```

5. **Use utility functions** to reduce code duplication:

php

```php
// Instead of manual checking
if (hasStatBonuses($item)) {
    // Show stats section
}
```

Dependencies

- These functions require the following to be available:
  - `SITE_URL` constant
  - `Database` class
  - Proper file structure in `/assets/img/`

File Structure Expected

```
/assets/img/
├── icons/           # Item icons (PNG format)
├── sprites/         # Monster/item sprites (PNG/GIF)
└── placeholders/    # Fallback images
    ├── 0.png        # Default item placeholder
    ├── sprite.png   # Default sprite placeholder
    └── monster.png  # Default monster placeholder
```

Migration from Old Functions

If you're updating from the old functions.php, replace:

Old Function → New Function

- `formatWeaponType()` → Use the updated `formatWeaponType()` (now handles all 18 types)
- `formatArmorType()` → Use the updated `formatArmorType()` (now handles all 20 types)
- `formatMaterial()` → Use `formatWeaponMaterial()`, `formatArmorMaterial()`, or `formatEtcitemMaterial()`
- `formatArmorGrade()` → Use `formatGrade()` or table-specific functions

Last updated: May 23, 2025

Version: 2.0