Moodle DX Update

Andrew Lyons
Principal Architect
Moodle LMS

Agenda

- **Changes to Moodle Versioning and Deprecation policies**
- **Upgrade Notes from Moodle 4.5**
- **Coding Style and related tooling**
- **Dependency Injection**
  - Clock
  - Hooks

Moodle Versioning

- **Changing *after* Moodle 4.5**
- **No change to:**
  - Frequency
  - Release cycle
  - LTS cycle
- **The release *after* an LTS will be a new *Series* version**
- **The last release in a series will be an LTS**

Moodle versioning

Moodle 4.5 LTS -- Under development

Moodle 5.0 -- New Series
Moodle 5.1
Moodle 5.2
Moodle 5.3 LTS -- Last in Series

Moodle 6.0 -- New Series
Moodle 6.1
Moodle 6.2
Moodle 6.3 LTS -- Last in Series

Rationale

- Clearer meaning of version numbers
  - Last release is *always* an LTS
- Aim to land biggest changes at the start of a new series
- Give more time for big new changes to stabilise
- More stable LTS releases
- More predictable change planning for partners and larger institutions

Deprecation Policies

Previous policy

Where possible:

Emit debugging and continue to work for FOUR major versions

Example:

Something deprecated in 4.1 will be removed in Moodle 4.5

Something deprecated in Moodle 4.5 will be removed in Moodle 5.3

New policy

Where possible:

Emit debugging and continue to work until the release after the *next* LTS

Example:

Something deprecated *before* Moodle 4.5 will be removed in Moodle 5.0

Something deprecated in Moodle 4.5 -> 5.2 will be removed in Moodle 6.0

Deprecated	Previous policy	New policy	
4.1	4.5 LTS	4.5 LTS	
4.2	5.0	5.0	New policy starts
4.3	5.1	5.0	
4.4	5.2	5.0	
4.5	5.3 LTS	**6.0**	
5.0	6.0	6.0	
5.1	6.1	6.0	
5.2	6.2	6.0	
5.3	6.3 LTS	**7.0**	

Rationale

- Reduce burden on developers
- Easier to work out when removals will occur
- Less frequent removals
- Encourage plugin developers to have one branch per Moodle Series

More examples

Changes to JS Minification

PHP Version support (MDLSITE-7677)

Questions..?

Developer Upgrade Notes

- Specifically upgrade.txt
- Intended to make it easier for plugin developers to discover changes which impact them
- Lots of them - 127 at last count
- Spread out across Moodle
- Lack consistency, and sorting, and standardisation
- Not clear what should be mentioned or where

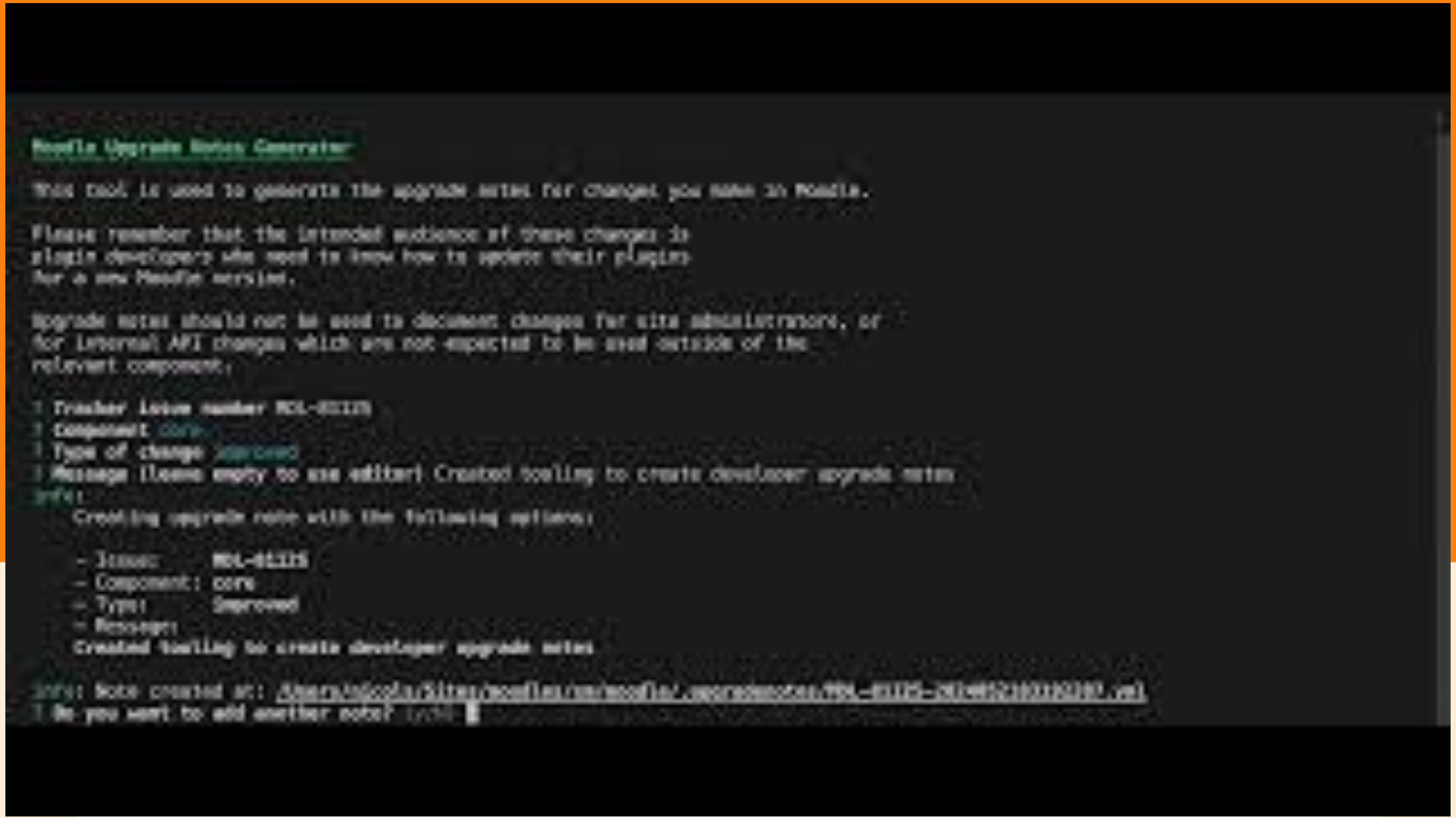
Developer Upgrade Notes (MDL-81125)

- **Impacts people contributing to Moodle *core* only**
- **For Moodle 4.5 onwards**
- **Move away from handwritten upgrade.txt files**
- **Use CLI tooling to write upgrade notes**
- **Generate Markdown files**
  - Central UPGRADING.md; *and*
  - Per-component UPGRADING.md
- **Collect specific information:**
  - Issue number
  - Type of change
  - Component

https://moodledev.io/general/development/upgradenotes

Rationale

- Improve discoverability of changes
- Link to the issue where a change was made
- Provide guidance on when, what, and why a change should be documented
- Allows better integrations with developer documentation
- Prevents merge and rebase conflicts for those making changes to busier core components


Moodle Upgrade Notes Generator
This tool is used to generate the upgrade notes for changes you make in Moodle.
Please remember that the intended audience of these changes is
plugin developers who need to know how to update their plugins
for a new Moodle version.
Upgrade notes should not be used to document changes for site administrators, or
for internal API changes which are not expected to be used outside of the
relevant component.
? Tracker issue number MDL-81135
? Component core
? Type of change improved
? Message (leave empty to use editor) Created tooling to create developer upgrade notes
info:
Creating upgrade note with the following options:
- Issue: MDL-81135
- Component: core
- Type: improved
- Message:
Created tooling to create developer upgrade notes
info: Note created at: /Users/stronk7/Sites/moodle.com/moodle/upgradenotes/MDL-81135-20240523101102.yml
? Do you want to add another note? (y/N)

Questions..?

Moodle coding style

- Updates to the moodle-cs rulesets for PHP_CodeSniffer
- Working towards deprecating and eliminating the need for moodle-local_codechecker
- Adding additional rules

https://moodledev.io/general/development/tools/phpcs

Dependency Injection

- **Available since Moodle 4.4**
- **Encourages writing to an Interface**
- **Allows you to swap out components**
- **Allows you to swap in mocked versions of classes for testing**
  - Hooks
  - Guzzle Client
  - Time
- **Automatically reset between tests**

https://moodledev.io/docs/4.5/apis/core/di

Dependency injection in legacy code

Fetching dependencies using the DI container

```
// Fetching an instance of the \core\http_client class outside of a class.
$client = \core\di::get(\core\http_client::class);

// Fetching an instance of a class which is managed using DI.
$thing = \core\di::get(my_thing::class);
```

Dependency injection in newer code

Injecting via the constructor

```php
class thing_manager {
    public function __construct(
        protected readonly \moodle_database $db,
    ) {
    }

    public function get_things(): array {
        return $this->db->get_records('example_things');
    }
}

// Fetching the injected class from legacy code:
$manager = \core\di::get(thing_manager::class);
$things = $manager->get_things();
```

Define dependencies of your class in constructor using type hints

When you fetch your class using DI, dependencies are resolved

Dependency injection in newer code

```
// Using it in a child class:
class other_thing {
    public function __construct(
        protected readonly thing_manager $manager,
    ) {
    }

    public function manage_things(): void {
        $this->manager->get_things();
    }
}
```

Dependencies are recursively resolved

Dependency injection in newer code

```
public function get_templates(
    ServerRequestInterface $request,
    ResponseInterface $response,
    mustache_template_source_loader $loader,
    string $themename,
    string $component,
    string $identifier,
): payload_response {
```

New Routing system aimed at Moodle 4.5 uses DI

MDL-81031

DI and Hooks

Use DI to dispatch a hook:

```
\core\di::get(\core\hook\manager::class)->dispatch($hook);
```

Allows you to mock the hook manager and include custom hook callbacks for testing

https://moodledev.io/docs/4.5/apis/core/hooks#dispatching-hooks

DI and Time

- New `\core\clock` implementation since Moodle 4.4
- Meets PSR-20: Clock
- Allows you to become a Time Lord in Unit Tests


POLICE PUBLIC CALL BOX
POLICE TELEPHONE
FREE
FOR USE OF
PUBLIC
PULL TO OPEN

DI and Time

```php
namespace mod_example;

class post {
    public function __construct(
        protected readonly \core\clock $clock,
        protected readonly \moodle_database $db,
    )

    public function create_thing(\stdClass $data): \stdClass {
        $data->timecreated = $this->clock->time();

        $data->id = $this->db->insert_record('example_thing', $data);

        return $data;
    }
}
```

```php
class my_test extends \advanced_testcase {
    public function test_create_thing(): void {
        // This class inserts data into the database.
        $this->resetAfterTest(true);

        $clock = $this->mock_clock_with_frozen();

        $post = \core\di::get(post::class);
        $posta = $post->create_thing((object) [
            'name' => 'a',
        ]);

        sleep(10);

        $postb = $post->create_thing((object) [
            'name' => 'a',
        ]);

        // The frozen clock keeps the same time.
        $this->assertEquals($postb->timecreated, $posta->timecreated);
```

Use a frozen clock

The `timecreated` will be the same because the post uses the `\core\clock`

Questions..?