Form API Revisited

The form constructor:

```
$myform = new myform_class(
    $action,// action attribute, autodetect default
    $customdata, // an array of additional data
    $method, // get or post
    $target, // deprecated in xHTML strict
    $attributes,  // HTML attributes
    $editable, // if not, no input fields shown
    $ajaxformdata // AJAX form data here
);
```

Form Elements Revisited

- Remember I said previously, only two required steps to add an element to a form;
  - Define the element - `$mform->addElement();`
  - Set the element's type with `$mform->setType()` by specifying the input's parameter type constants, such as `PARAM_TEXT`

We will now look at other methods.

Form Elements API (cont)

`$mform->addHelpButton()` method - specify some text for the help popup/tooltip and expects three arguments, namely:

- The name of the form element for which the help button is to be added. This is not a button, but rather a help icon that is displayed next to the element's text;
- The identifier for the help string title and contents. Both the title and the contents should be defined in the plugin's strings file. Moodle gets the title from the string with the identifier specified and the contents from the identifier specified with '_help' appended. For example, if the identifier is the username, the title value is `$string['username']` and the help contents is `$string['username_help'];`
- The plugin/component's name.

Form Elements API (cont)

`$mform->setDefault()` – Usually, the default value of the elements is set by the `$this->set_data()` method, but this requires that the form fields mirror the database record field names.

If not, globals or the data in the `$customdata` parameter passed into the form constructor be used to set the field default value. The custom data is accessed via the `$this->_customdata` variable e.g.

```
$mform->setDefault('noimages', $this->_customdata['noimages']);
```

Form API (cont)

`$mform->addRule()` - allows you to specify rules including:

- required
- maxlength
- email
- alphanumeric
- numeric

Form Elements API (cont)

The `$mform->addRule()` method requires at least three parameters, but takes a total of seven.

- The element's name
- The error message to display if the input is invalid – usually obtained via a `get_string()` call
- The rule type e.g. required
- Optionally, the data required by the rule, such as the maximum length for the maxlength rule
- Optionally, to perform validation, either 'server', which is the default, or 'client'
- Optionally, a boolean value for client-side validation, determining whether the form should be reset if there is an error – defaults to false
- Whether to force the rule to be applied, even if the target form element does not exist – defaults to false

See the Moodle documentation's addRule paragraph on the 'formslib.php_Form_Definition' page (https://docs.moodle.org/dev/lib/formslib.php_Form_Definition#addRule).

Form Elements API (cont)

Loosely related to the rules are two other useful methods:

- `$mform->disabledIf()`
- `$mform->hideif()`

which both accept the same parameters that allow you to conditionally hide or disable any group or individual element depending on conditions provided as one of the parameters.

Form Elements API (cont)

`$mform->createElement()` is usually used to group form elements which will have a single label and are included on one line in the form. Each created element *reference* is added to an array that is then passed to `$mform->addGroup()` method.

See https://docs.moodle.org/dev/Form_API#Form_elements

Form API (cont)

A special `moodle_form` method called as

```
$this->add_action_buttons()
```

adds action buttons to the form. The function can take up to two parameters:

- A boolean parameter to specify whether to display a cancel button – default is true
- The text for the submit button and the default is get_string('savechanges') from the core strings

mod_moodleform extensions

- `$this->standard_intro_elements()`: this adds an editor to allow the user to enter a description and, since we have said we support the `FEATURE_MOD_INTRO` feature, we should include this field;
- `$this->standard_coursemodule_elements()`: all modules should include this in the form, as this collects the standard data for modules. In short, it adds the following sections to the form:
  - Restrict access
  - Tags
  - Competencies
- `$this->add_action_buttons()`: this adds the submit buttons with the option to return to the course or to display the module or to cancel the whole operation. Again, all modules should use this functionality in order to be consistent with other modules.

Defining Elements

Normally, there are several steps to defining an element for a form:

1. First add the element - `$mform->addElement()`
2. Set the element's type with `$mform->setType()`
3. Optionally, specify some text for the help popup/tooltip using `$mform->addHelpButton()`
4. Optionally, set the elements default value using `$mform->setDefault()`

Form API Revisited (cont)

The most useful form methods are: (but, remember, there are many more, so it is worth having a look at the `moodleform` class code):

- `display(), render()` – print or get the form HTML
- `set_data()` – apply existing database table data values to the form elements; the fields must match for this to work
- `validation()` – function to validate the submitted data; errors mean the form is re-displayed with errors highlighted
- `get_data()` – a special wrapper method to get the submitted data or null if no data has been submitted or validation fails
- `is_cancelled()` – checks if the form has been cancelled
- `is_submitted()` – check if the form has been submitted.

Form API Revisited (cont)

Generally, the process is as follows:

1. Initiate the form, e.g. `$myform = new myformclass()`
2. Optionally read data from the database and apply it to the form using the `set_data()` method.
3. Check if the form has been cancelled using the `is_cancelled()` method and act accordingly – usually by redirecting the user.
4. Check if data has been submitted with the `get_data()` method and process if necessary.
5. If not cancelled or submitted, display the form with the `display()` method.

See the Moodle documentation's 'lib/formslib.php Usage' page (https://docs.moodle.org/dev/lib/formslib.php_Usage) for another description of this process.