+++ title = “Error handling” +++

Error handling

This guide explains how to handle errors in plugins.

Provide usable defaults

Allow the user to learn your plugin in small steps. Provide a useful default configuration so that:

- The user can get started right away.
- You can avoid unnecessary error messages.

For example, by selecting the first field of an expected type, the panel can display a visualization without any user configuration. If a user explicitly selects a field, then use that one. Otherwise, default to the first field of type `string`:

```
const numberField = frame.fields.find((field) =>
  options.numberFieldName ? field.name === options.numberFieldName : field.type === FieldTyp
);
```

Display error messages

To display an error message to the user, `throw` an `Error` with the message you want to display:

```
throw new Error('An error occurred');
```

Grafana displays the error message in the top-left corner of the panel.

{{< figure src=“/static/img/docs/panel_error.png” class=“docs-image–no-shadow” max-width=“850px” >}}

Avoid displaying overly-technical error messages to the user. If you want to let technical users report an error, consider logging it instead.

```
try {
  failingFunction();
} catch (err) {
  console.error(err);
  throw new Error('Something went wrong');
}
```

> **Note:** Grafana displays the exception message in the UI as written, so we recommend using grammatically correct sentences. For more information, refer to the Documentation style guide.

Here are some examples of situations where you might want to display an error to the user.

Invalid query response

Users have full freedom when they create data source queries for panels. If your panel plugin requires a specific format for the query response, then use the panel canvas to guide the user.

```
if (!numberField) {
  throw new Error('Query result is missing a number field');
}

if (frame.length === 0) {
  throw new Error('Query returned an empty result');
}
```