Using Context With ACF Blocks

- Creating Block Context
  - Example block that provides context
  - Example block that consumes context
- Displaying Context Values in ACF Blocks
- Conclusion

ACF Blocks opens the door to more creative building possibilities. One example we previously explored was using InnerBlocks and parent/child relationships, where we nested ACF Blocks inside of each other. Nesting your ACF Blocks organizes your block structure and codebase while allowing for a refined editorial experience.

Ready to dive into the code?

You can dive right in by downloading the **tt4child.zip**! Be sure to have the TwentyTwentyFour theme installed as well as ACF PRO.

ACF 6.0 introduced support for WordPress's block context. Block context enables higher-level blocks, such as parent blocks, to supply values that descendant blocks within the same structure can utilize. This functionality permits direct child blocks and blocks further down the hierarchy to receive values from a parent block. This process occurs without the necessity for embedding these values directly into the code or establishing a direct link between the parent and descendant blocks.

It is critical to note that block context only flows *downward* from parent blocks, and it's not possible to share data upward from child blocks to their parent.

Creating Block Context

Block context is defined in the block's `block.json`, just like other block properties and attributes. The relationship of block context relies on there being a block that is the **provider** and a block that

is the **consumer** of the providing block's context.

The **provider** block defines `providesContext` in its `block.json`:

Example block that provides context

```
{
    ...rest of block.json

    "providesContext": {
        "acf/fields": "data"
    },

    ...rest of block.json
}
```

And the **consumer** block defines the `usesContext` in *its* `block.json`:

Example block that consumes context

```
{
    ...rest of block.json

    "usesContext": ["acf/fields"],

    ...rest of block.json
}
```

Displaying Context Values in ACF Blocks

Once you have established a binding context relationship between your ACF Blocks, you can reference values from the parent block in any nested block by checking and outputting the value in the child block's template.

```
<?php echo $context['acf/fields']['an_example_field']; ?>
```

One thing to note: ACF Blocks use field keys in the block editor, and block data is only hydrated to field names upon saving. Because of this, you should use `$context['acf/fields']['field_key']` to get access to the data inside your template if it exists, before defaulting back to `$context['acf/fields']['field_name']` if it doesn't. This supports both backend and frontend output of context data.

Also, whenever the providing parent block's field value is updated in the editor, the child block that consumes the parent's context will be automatically re-rendered.

You should also be aware that sharing ACF data with your child blocks will result in sending the complete ACF field data back to the server for rendering previews. This may have performance implications for your blocks in the editor.

Also, ACF automatically passes the `postId` and `postType` down through block context by default, which expedites nesting your ACF Block in the Query Loop block.

https://www.advancedcustomfields.com/wp-content/uploads/2024/01/ACF-block-context-demo.mp4

Conclusion

In summary, ACF Blocks introduces a versatile and powerful way to extend the functionality of the WordPress editor, offering a rich set of features that enhance both the developer and editor experiences. By leveraging InnerBlocks and parent/child relationships, you can structure your content with greater flexibility and precision. The introduction of block context in ACF 6.0 further amplifies this by allowing data to flow seamlessly between blocks, removing the need for direct code embeddings or explicit relationships.

This advancement simplifies the development process and opens up a myriad of creative possibilities for content management and presentation. Whether you are building complex layouts, crafting unique editorial experiences, or providing dynamic content relationships, ACF Blocks equipped with block context capabilities are an invaluable tool in your WordPress development arsenal.

Remember, while the power of ACF Blocks and block context brings great flexibility, it's important to consider performance implications and the best practices for data management to ensure your site remains fast and responsive. With the right approach, ACF Blocks can transform the way you think about and build with WordPress.

Download the **tt4child.zip** and dive into the code to see firsthand how ACF Blocks can revolutionize your WordPress projects. Happy coding!

□□□ #1
□ □□□ □□□ 2026-08-29 15:44:03 UTC
□ □□□ □□□ 2026-08-29 15:44:03 UTC