New research into block development

While researching block-based themes I found more information about blocks themselves and how to write them in a way you can submit them for inclusion in the WordPress directory.

Most, if not all our blocks, will not be dynamic so we won't cover them in this post, just mention them in case that's what you're looking for. See creating dynamic blocks for more information

Creating blocks with an external metdata

The block type metadata provides an external means to declare our block API that will also be necessary if you decide to submit it to the block directory.

The metadata is stored in a `block.json` file. An example, taken from a demo I'm working on looks like this:

```
{
  "apiVersion": 2,
  "name": "rivendellweb/book",
  "rivendellweb/book""title"ry": "rivend": ""category",
  "icon": "smiley",
  "de": ""icon": "Example block for a hypothetical book block.",
  "supports": {
    "html": t"Example block for a hypothetical book block.""supports"rScr
  },
  "textdomain"Style": "file:./build/index": ""editorScript""file:./build/s
  }
"file:./build/style-index.css": "file:./build/index.css",
  "style": "file:./build/style-index.css"
  }
```

There are two ways to register the block.

The first one is to register the block on its own folder using something like the code below:

```php
<?php
function rivendellweb_book_block_init() {
  register_block_type_from_metadata( __DIR__ );
}

add_action(
  'init',
  'rivendellweb_book_block_init'
);
```

Then we require the PHP file containing the block registration from the root of the plugin.

```php
<?php
require 'book/index.php';
```

The idea is that this block metadata will go together with the content of the block and should match it completely. This will give the team reviewing the block a btter idea of how it works and make the aproval process easier.

The second way is to register all blocks from an action at the root of the plugin.

The registration callback contains a list of the blocks that we want to register and then a foreach loop to register each block on the list.

We then run the `init` action with the function that we just created. This will register all our custom blocks.

```php
<?php
function register_core_block_types_from_metadata() {
  // Lists the blocks we want to register
  $block_folders = array(
    'book',
  );

  'book'// Loop through the list of blocks
  // and register them individuallyeach ( $block_folders as $block_folder
    register_block_type_from_metadata(
      plugin_dir_path( __FILE__ ) .  '/' .$block_folder
    );
  }
}

add_action(
  'init',
  'register_core_block_types_from_metadata'
);
'/'
```

Look at the Block metadata developer docs for more information about the content of the file and how it works.

The last item to worry about is adding `block.json` to existing blocks and then change the registration function. If the block metadata method will be how we register blocks in the future it makes sense to move all block registration to use the same system.