Gutenberg Message Block (Manual Setup)

This guide shows you how to manually build a **custom Gutenberg block** with editable settings (via sidebar panel), using only **vanilla JavaScript and PHP** — no React/JSX build tools required.

We'll walk step-by-step through creating a block that lets the user enter a **custom message** via the block settings sidebar, and displays it both in the editor and on the front end.

✨Features

- Custom block titled **Mgb Message Widget**
- Editable message input from sidebar controls (InspectorControls)
- Server-side rendering via PHP
- Built without build tools (no Webpack, no JSX)

File Structure

```
my-plugin/
├── widget/
│   ├── widget-blocks.js         # Block JS (manual)
│   └── widget-block.php          # PHP logic for block registration
```

🐐 Step 1: PHP Setup

Create the file: `widget/widget-block.php`

PHP Code (register + render block)

```php
<?php
if (!defined('ABSPATH')) {
    exit;
}

class Widget_Block
{
    public function __construct()
    {
        add_action('init', [$this, 'register_blocks']);
```

```
        add_action('enqueue_block_editor_assets', [$this, 'load_assets']);
    }

    public function register_blocks()
    {
        register_block_type('mgb/message-block', [
            'render_callback' => [$this, 'render_message_block'],
            'attributes' => [
                'message' => [
                    'type' => 'string',
                    'default' => 'Hello World!'
                ]
            ]
        ]);
    }

    public function render_message_block($attributes, $content)
    {
        $msg = $attributes['message'] ? esc_html($attributes['message']) :
'Hello World!';
        ob_start();
        ?>
        <div class="mgb_message_container">
            <p><?php echo $msg; ?></p>
        </div>
        <?php
        return ob_get_clean();
    }

    public function load_assets()
    {
        wp_enqueue_script(
            'mgb-blocks-js',
            MGB_URL . 'widget/widget-blocks.js',
            ['wp-blocks', 'wp-element', 'wp-editor', 'wp-components', 'wp-
i18n'],
            '1.0.0',
            true
        );
    }
}

new Widget_Block();
```

⚠️Make sure `MGB_URL` is defined elsewhere, usually in your main plugin file:

```
define('MGB_URL', plugin_dir_url(__FILE__));
```

🌳Step 2: JavaScript Block Registration

Create the file: `widget/widget-blocks.js`

JavaScript Code (no JSX)

```
wp.blocks.registerBlockType("mgb/message-block", {
  title: "Mgb Message Widget",
  icon: "dashicons-buddicons-pm",
  category: "widgets",
  attributes: {
    message: {
      type: "string",
      default: "Hello World!",
    },
  },
  edit: (props) => {
    const InspectorControls = wp.blockEditor
      ? wp.blockEditor.InspectorControls
      : wp.editor.InspectorControls;
    const PanelBody = wp.components.PanelBody;
    const inputStyle = {
      width: "100%",
      marginBottom: "12px",
      boxSizing: "border-box",
    };

    return wp.element.createElement(
      wp.element.Fragment,
      {},
      wp.element.createElement(
        InspectorControls,
        {},
        wp.element.createElement(
          PanelBody,
          {
            title: "Message Controls",
            initialOpen: true,
          },
          wp.element.createElement(
            "div",
            { style: { marginBottom: "16px" } },
```

```
            wp.element.createElement(
              "label",
              { htmlFor: "mgb-message" },
              "Message Input"
            ),
            wp.element.createElement("input", {
              id: "mgb-message",
              type: "text",
              value: props.attributes.message,
              onChange: function (e) {
                props.setAttributes({ message: e.target.value });
              },
              placeholder: "Enter your message...",
              style: inputStyle,
            })
          )
        )
      ),
      wp.element.createElement(
        "div",
        {},
        wp.element.createElement("p", {}, props.attributes.message)
      )
    );
  },
  save: () => {
    return null; // Server-side render only
  },
});
```

Editor View Output

The message is editable in the sidebar. The live preview updates in the editor (middle of block).

Frontend Output (After Save)

HTML Output:

```
<div class="mgb_message_container">
    <p>Hello World!</p>
</div>
```

The content is rendered using `render_callback` in PHP — safe and dynamic.

🦘 Summary

Step	What You Did	Tool
1	Created PHP block registration	WordPress
2	Wrote JS block without JSX	Vanilla JS
3	Added InspectorControls for input	Gutenberg
4	Rendered dynamic output from PHP	Server-side

🐻‍❄️ Bonus Ideas

- Add color or font controls
- Store multiple messages
- Add `block.json` to move toward modern registration

Let me know if you want a JSX/Babel/Webpack version!

no only manual version