Block Development Workshop

Josh Pollock

About

Josh Pollock

- VP Engineering Experience: Saturday Drive
    - Ninja Forms, Caldera Forms, SendWP, Ninja Shop & more!
- Formerly Co-Owner & Lead Developer: CalderaWP
- WordPress Core Contributor
- Wrote a lot of tutorials about WordPress development

Workshop Overview

- Structuring a block plugin
- Basic Static Block
    - Block anatomy
- Editable Block
    - Block attributes
    - Using WordPress components
- Whatever y'all want to do next
    - Shortcode replacement, plugin sidebars, state management, testing, dynamic blocks, isomorphic blocks,

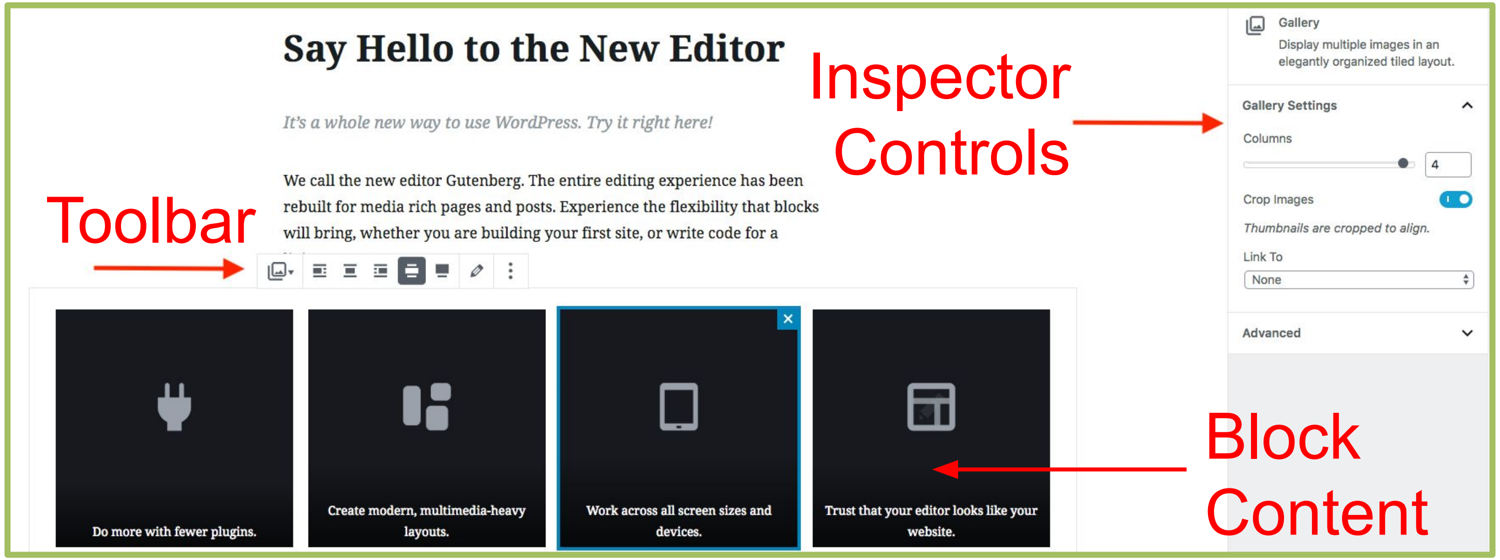
Cut & Paste From The README Like A Professional

github.com/shelob9/block-worskshop/

How do I get started? Where does everything go?

Hi Roy Block

The Anatomy of Using a Block



Say Hello to the New Editor

It's a whole new way to use WordPress. Try it right here!

We call the new editor Gutenberg. The entire editing experience has been rebuilt for media rich pages and posts. Experience the flexibility that blocks will bring, whether you are building your first site, or write code for a

Do more with fewer plugins.

Create modern, multimedia-heavy layouts.

Work across all screen sizes and devices.

Trust that your editor looks like your website.

Gallery

Display multiple images in an elegantly organized tiled layout.

Gallery Settings

Columns: 4

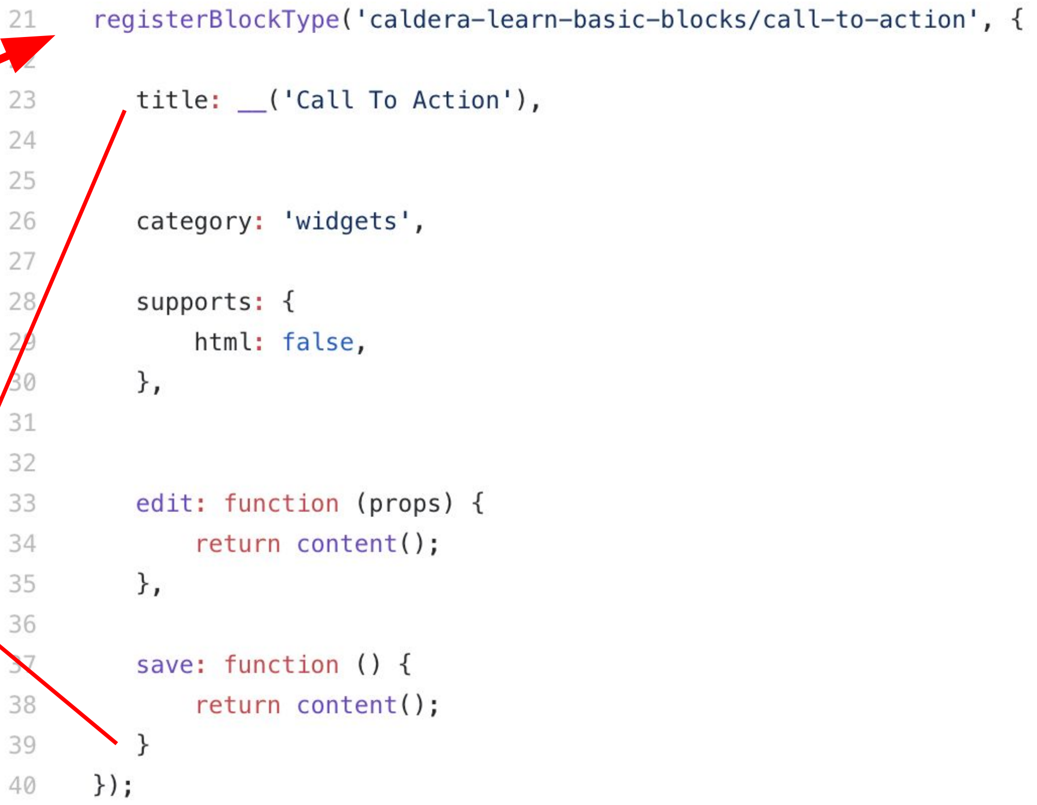
Crop Images

Thumbnails are cropped to align.

Link To: None

Advanced

The Anatomy of Coding a Block



```
registerBlockType('caldera-learn-basic-blocks/call-to-action', {

    title: __('Call To Action'),


    category: 'widgets',

    supports: {
        html: false,
    },


    edit: function (props) {
        return content();
    },

    save: function () {
        return content();
    }
});
```

registerBlockType()

registerBlockType()
Settings

It Can Get Complex

```
    edit({attributes, setAttributes, isSelected}) {
        const {
            text,
            linkText,
            link,
            align,
            textColor,
            backgroundColor,
            linkColor
        } = attributes;


        const onChangeText = (text) => setAttributes({text});
        const onChangeLinkText = (linkText) => setAttributes({linkText});
        const onChangeLink = (link) => setAttributes({link});
        const onChangeAlign = (align) => setAttributes({align});
        const onChangeTextColor = (textColor) => setAttributes({textColor});
        const onChangeBackgroundColor = (backgroundColor) => setAttributes({backgroundColor});
        const onChangeLinkColor = (linkColor) => setAttributes({linkColor});


        if (isSelected) {
            return (
                <div>
                    <InspectorControls>
                        <TextControl
                            label={'Link Text'}
                            value={linkText}
                            onChange={onChangeLinkText}
                        />
                        <TextControl
                            label={'Link Url'}
                            value={link}
                            onChange={onChangeLink}
                        />
                        <ColorControl
                            label={'Text Color'}
                            onChange={onChangeTextColor}
                            value={textColor}
                        />
                        <ColorControl
```

```
            align,
            textColor,
            backgroundColor,
            linkColor
        } = attributes;


        const onChangeText = (text) => setAttributes({text});
        const onChangeLinkText = (linkText) => setAttributes({linkText});
        const onChangeLink = (link) => setAttributes({link});
        const onChangeAlign = (align) => setAttributes({align});
        const onChangeTextColor = (textColor) => setAttributes({textColor});
        const onChangeBackgroundColor = (backgroundColor) => setAttributes({backgroundColor});
        const onChangeLinkColor = (linkColor) => setAttributes({linkColor});


        if (isSelected) {
            return (
                <div>
                    <InspectorControls>
                        <TextControl
                            label={'Link Text'}
                            value={linkText}
                            onChange={onChangeLinkText}
                        />
                        <TextControl
                            label={'Link Url'}
                            value={link}
                            onChange={onChangeLink}
                        />
                        <ColorControl
                            label={'Text Color'}
                            onChange={onChangeTextColor}
                            value={textColor}
                        />
```

```
                        <ColorControl
                            label={'Link Color'}
                            onChange={onChangeLinkColor}
                            value={linkColor}
                        />
                        <ColorControl
                            label={'Background Color'}
                            onChange={onChangeBackgroundColor}
                            value={backgroundColor}
```

Start Simple

Write Code Now

Install Some Stuff

- Create package.json:
- Install WordPress scripts
  - npm i @wordpress/scripts
- Update "scripts" in package.json
  - Copy from README
- Ignore directories in .gitingore

Add Files & (Don't) Configure webpack & Babel

- Create src/index.js
- Put some JavaScript in there.
- Compile it to make sure everything works.
  - npm run build
  - You should see built JavaScript
- Be impressed by how simple webpack and Babel are to use :)
- The one PHP file!

Write Block Now

Static, Non-editable Block

- Open src/index.js
- Write Some JavaScript:
    - Import dependencies
    - registerBlockType()
    - Edit Callback
    - Save Callback

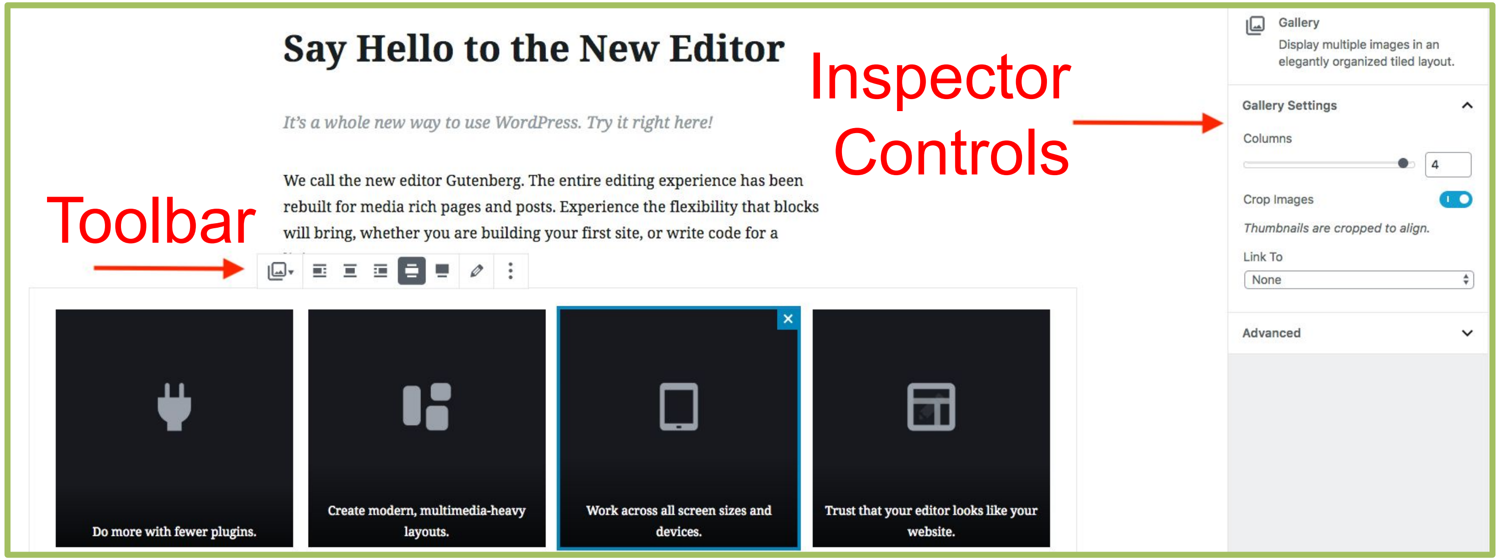
Transpile Is A Cool Word!

- For manual testing while developing
  - npm run start
- To ship for production
  - npm run build

How do I make controls? Where Does Data Go?

Editable Block

The Toolbar and Inspector Controls



Say Hello to the New Editor

It's a whole new way to use WordPress. Try it right here!

We call the new editor Gutenberg. The entire editing experience has been rebuilt for media rich pages and posts. Experience the flexibility that blocks will bring, whether you are building your first site, or write code for a

Do more with fewer plugins.

Create modern, multimedia-heavy layouts.

Work across all screen sizes and devices.

Trust that your editor looks like your website.

Gallery

Display multiple images in an elegantly organized tiled layout.

Gallery Settings

Columns: 4

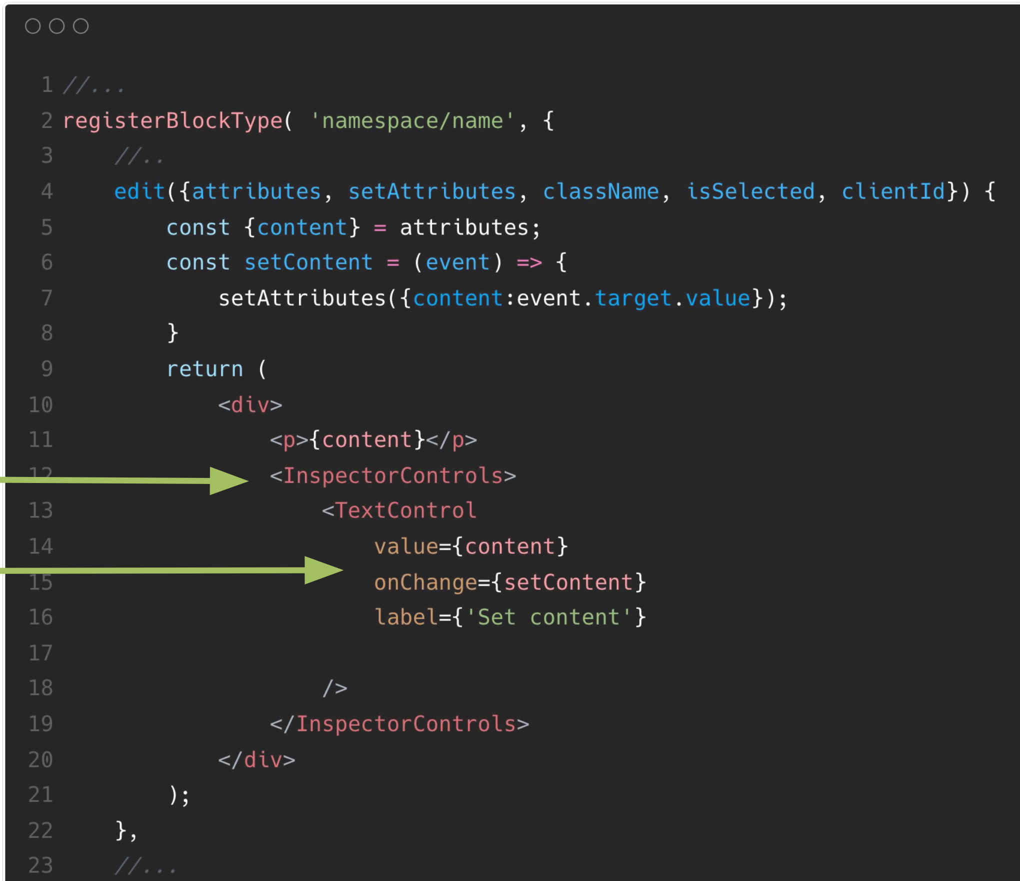
Crop Images

Thumbnails are cropped to align.

Link To: None

Advanced

Inspector Controls



```
//...
registerBlockType( 'namespace/name', {
    //..
    edit({attributes, setAttributes, className, isSelected, clientId}) {
        const {content} = attributes;
        const setContent = (event) => {
            setAttributes({content:event.target.value});
        }
        return (
            <div>
                <p>{content}</p>
                <InspectorControls>
                    <TextControl
                        value={content}
                        onChange={setContent}
                        label={'Set content'}

                    />
                </InspectorControls>
            </div>
        );
    },
    //...
```

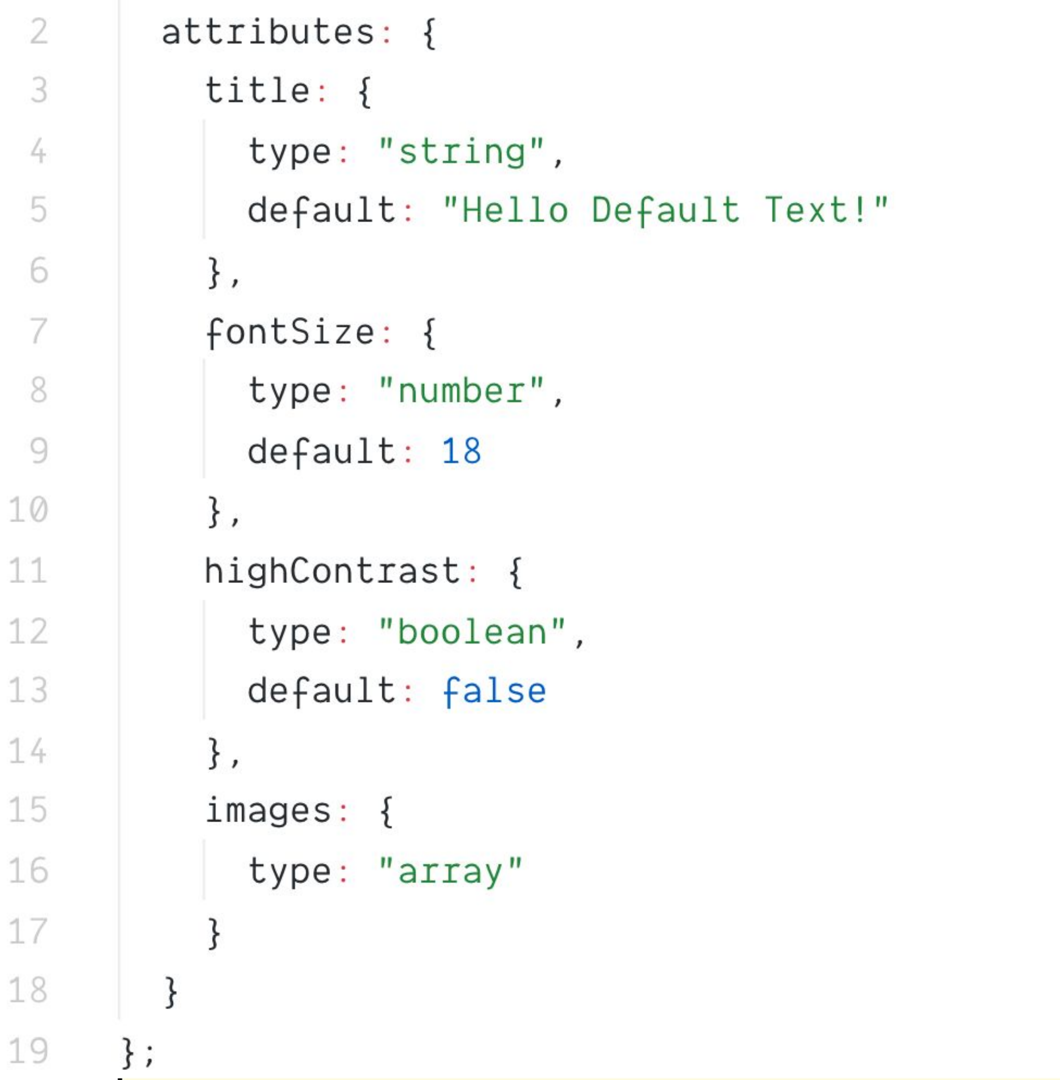
Put components inside of InspectorControls component

Where Do I Put Data?

Attributes

Types of Attributes

- Strings
- Numbers
- Booleans
- Arrays



```
  attributes: {
    title: {
      type: "string",
      default: "Hello Default Text!"
    },
    fontSize: {
      type: "number",
      default: 18
    },
    highContrast: {
      type: "boolean",
      default: false
    },
    images: {
      type: "array"
    }
  }
};
```

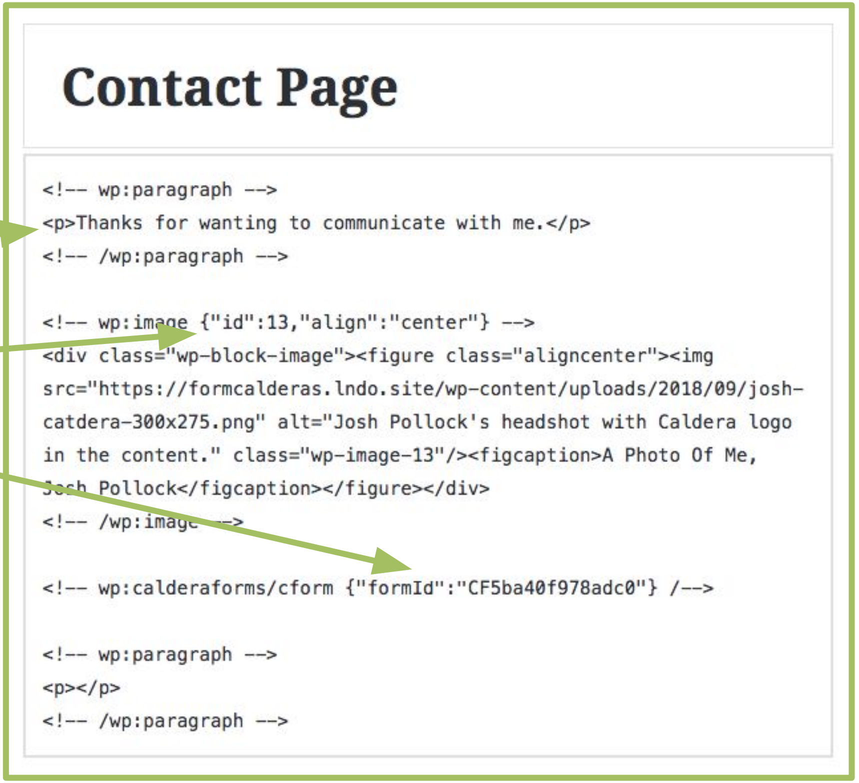
Attribute Storage Methods

- **Default** - Serialized in HTML comments
- **Sourced** - Stored as part of the HTML content
- Post Meta - Stored in database as post meta

Default Attributes Serialized in Comments

Store in:

- HTML
- HTML comments
  - For shortcode-like blocks
- Post Meta *



Contact Page

```
<!-- wp:paragraph -->
<p>Thanks for wanting to communicate with me.</p>
<!-- /wp:paragraph -->

<!-- wp:image {"id":13,"align":"center"} -->
<div class="wp-block-image"><figure class="aligncenter"><img
src="https://formcalderas.lndo.site/wp-content/uploads/2018/09/josh-
catdera-300x275.png" alt="Josh Pollock's headshot with Caldera logo
in the content." class="wp-image-13"/><figcaption>A Photo Of Me,
Josh Pollock</figcaption></figure></div>
<!-- /wp:image -->

<!-- wp:calderaforms/cform {"formId":"CF5ba40f978adc0"} /-->

<!-- wp:paragraph -->
<p></p>
<!-- /wp:paragraph -->
```

Write More Code Now Please

Make Block Editable

- Install new dependencies:
  - npm i @wordpress/editor @wordpress/components
- Import TextControl from @wordpress/components
  - Use for editting message
- Import InspectorControls from @wordpress/editor

Actually Make It Editable

- Add message attribute
- Get current value of message from attributes prop passed to save and edit.
- Create update function for message using setAttributes prop passed to edit callback.
- Display current value of message in edit and save callbacks.
- Use TextControl to created edit interface for block.
- Wrap the control in InspectorControls so it goes in the inspector controls.

What else do you want to know? How much time do we have left?

Cooler Blocks

Taking Gutenberg Further

- Shortcode to block conversion
- Plugin sidebars
- Block testing
- Isomorphic blocks
- React skills
- Redux Data Module
- Other frameworks besides React

Thank You!

Josh Pollock

- Josh412
- JoshPress.net
- SaturdayDrive.io

Slides And Links:

- github.com/Shelob9/block-worskshop/