Dynamic Blocks… for the win!

Customize Gutenberg without living in fear of validation errors.

Hi, I’m Joni!

- **Nerd Life:** Developer @ Georgetown University
- **Home Life:** Mom to an adorable 3 year old
- **Hobby Life:** Runner, reader, terrible ukulele player

Expectation setting

- Technical talk for developers
- Lots of talk about JS and PHP
- Knowledge of Gutenberg block creation is helpful

How this all started

- Migrating from Drupal to WordPress
- Decided to embrace Gutenberg in its alpha stage
- Learned React / Gutenberg API on the go

A simple example

Once upon a time...

Jordan, who manages the library site, asks for a website block that can display a book's title and author.

No problem, you say! We can create a custom Gutenberg block for that.

Book block!

Let's create the block, the standard way.

Step 1

Register the block.

```
import BookEdit from './edit.js';
import BookSave from './save.js';

const Book = ( () => {

  const { registerBlockType } = wp.blocks;

  registerBlockType( 'my/book', {
    title: 'Book',
    description: 'A simple book block.',
    category: 'text',
    icon: 'book-alt',

    attributes: {
      title: { type: 'text' },
      author: { type: 'text' }
    },
```

Step 2

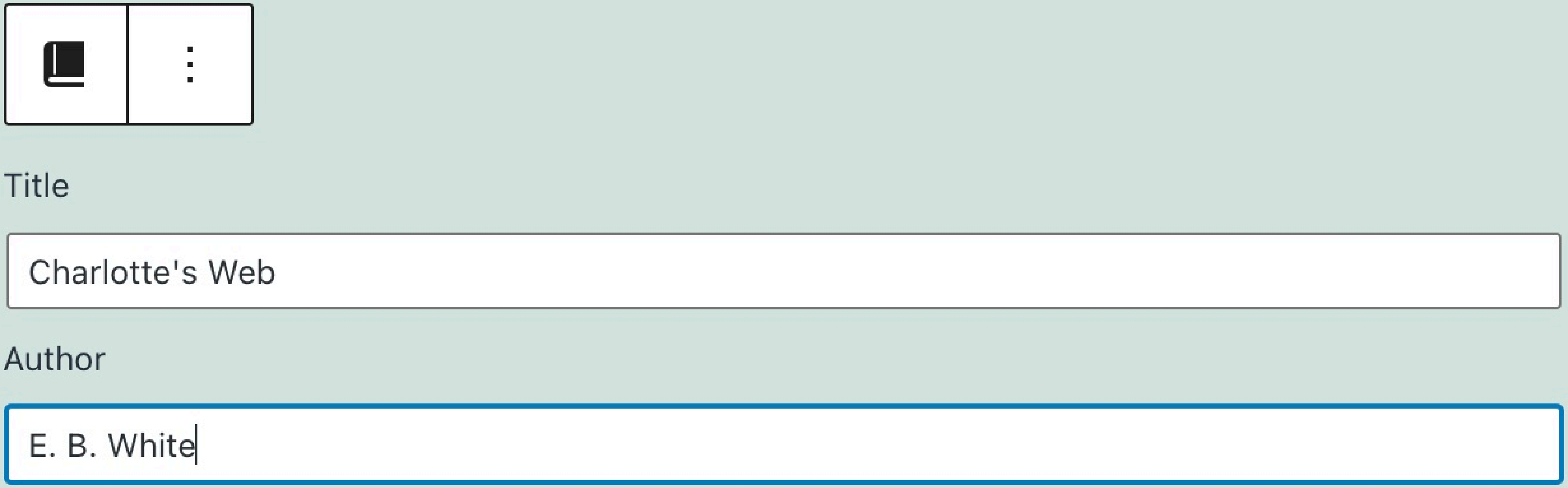
Add the editor UI.

```
const BookEdit = ( props ) => {

  // Get WP packages.
  const { TextControl } = wp.components;
  const { Fragment } = wp.element;

  // Get block properties.
  const { setAttributes } = props;
  const { title, author } = props.attributes;

  // Set on change callbacks.
  const onChangeTitle = ( value ) => {
    setAttributes( { title: value } )
  };

  const onChangeAuthor = ( value ) => {
    setAttributes( { author: value } )
  };
```

Step 3

Create the front-end markup.

```
const BookSave = ( props ) => {

  const { title, author } = props.attributes;

  return (
    <div class="my-book-block">
      <p>{ title }</p>
      <p>{ author }</p>
    </div>
  );

}

export default BookSave;
```


Title

Charlotte's Web

Author

E. B. White

My Favorite Books

Charlotte’s Web

E. B. White

Our story continues

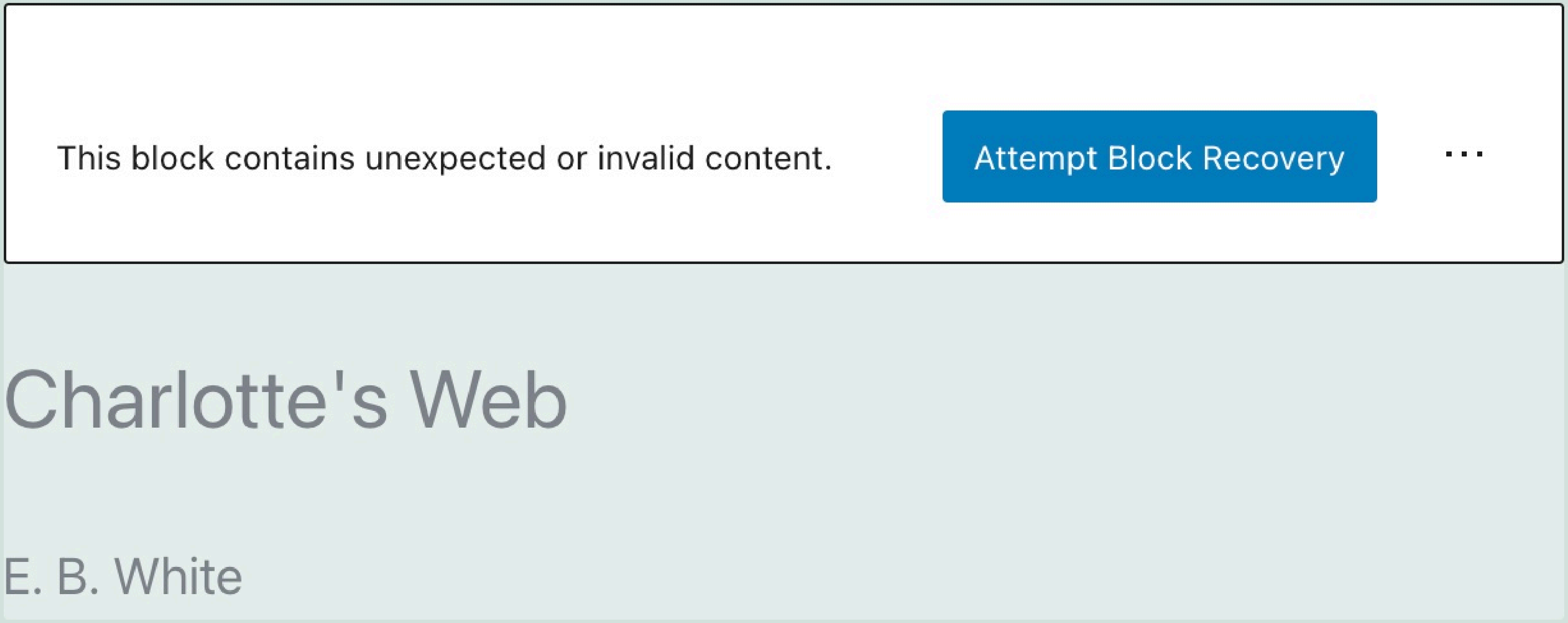
Jordan loves the new Book block! However, it looks a bit plain. Can we change the title so that it is a header instead of a paragraph?

Sure, you say! Easy peasy.

Book block, v2

Update the markup in the `save()` function to use a `<h3>` tag instead of a `<p>` tag.

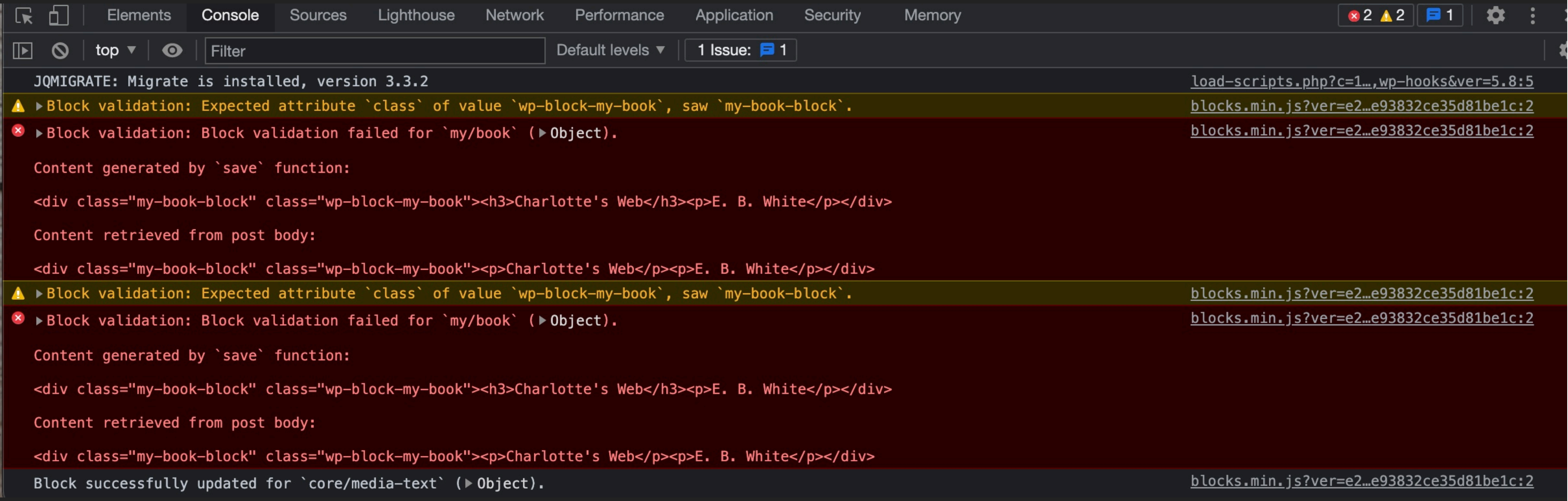
```
return (
  <div class="my-book-block">
    <h3>{ title }</h3>
    <p>{ author }</p>
  </div>
)
```


This block contains unexpected or invalid content.

Charlotte's Web

E. B. White

Block Validation Error!


Elements Console Sources Lighthouse Network Performance Application Security Memory

top ▼ Filter Default levels ▼ 1 Issue: 1

```
JQMIGRATE: Migrate is installed, version 3.3.2                                  load-scripts.php?c=1…,wp-hooks&ver=5.8:5
▶Block validation: Expected attribute `class` of value `wp-block-my-book`, saw `my-book-block`.   blocks.min.js?ver=e2…e93832ce35d81be1c:2
▶Block validation: Block validation failed for `my/book` (▶Object).                blocks.min.js?ver=e2…e93832ce35d81be1c:2

Content generated by `save` function:

<div class="my-book-block" class="wp-block-my-book"><h3>Charlotte's Web</h3><p>E. B. White</p></div>

Content retrieved from post body:

<div class="my-book-block" class="wp-block-my-book"><p>Charlotte's Web</p><p>E. B. White</p></div>
▶Block validation: Expected attribute `class` of value `wp-block-my-book`, saw `my-book-block`.   blocks.min.js?ver=e2…e93832ce35d81be1c:2
▶Block validation: Block validation failed for `my/book` (▶Object).                blocks.min.js?ver=e2…e93832ce35d81be1c:2

Content generated by `save` function:

<div class="my-book-block" class="wp-block-my-book"><h3>Charlotte's Web</h3><p>E. B. White</p></div>

Content retrieved from post body:

<div class="my-book-block" class="wp-block-my-book"><p>Charlotte's Web</p><p>E. B. White</p></div>
Block successfully updated for `core/media-text` (▶Object).                      blocks.min.js?ver=e2…e93832ce35d81be1c:2
```

Block validation

Gutenberg likes to make sure the block's front-end code is what it expects it to be.

Validation process

- Pre-render the block using the current `save()` function and block attribute data.
- Compare pre-rendering to the code in the database.
- If it matches, **Success**!
- If it does not match, look for known deprecations.

In the database

```
<!-- wp:my/book {"title":"Charlotte's Web",
"author":"E. B. White"} -->
<div class="my-book-block" class="wp-block-my-book">
<p>Charlotte's Web</p>
<p>E. B. White</p>
</div>
<!-- /wp:my/book -->
```

Remember

This is our `save()` markup:

```
<div class="my-book-block">
  <h3>{ title }</h3>
  <p>{ author }</p>
</div>
```

Deprecations

Deprecations are previous versions of the block that WordPress can use to validate updated blocks.

```
const BookDeprecated = [
  {
    attributes: {
      title: { type: 'text' },
      author: { type: 'text' }
    },

    save: ( props ) => {
      const { title, author } = props.attributes;

      return (
        <div class="my-book-block">
          <p>{ title }</p>
          <p>{ author }</p>
        </div>
      );
    }
  }
```

Every time you change the expected output, you need a new deprecation object item.

Dynamic blocks to the rescue!

Definition:

Static block

- Block is registered in the JS
- Markup is created by the JS `save()` function
- Markup is saved in the DB as part of the post content

Definition:

Dynamic block

- Block is registered in both the JS and PHP
- Only block attribute data is saved to the DB
- No markup is saved to the DB
- Markup is rendered by the PHP

Book block, take 2

Dynamic edition

Step 1

Register and render the block in PHP.

```php
class Book {

  public function __construct() {
    add_action( 'init', [$this, 'register'] );
  }

  public function register(): void {
    register_block_type( 'my/book', [

      'attributes' => [
        'title'  => [ 'type' => 'string' ],
        'author' => [ 'type' => 'string' ]
      ],

      'render_callback' => [ $this, 'render' ]
    ] );
  }
```

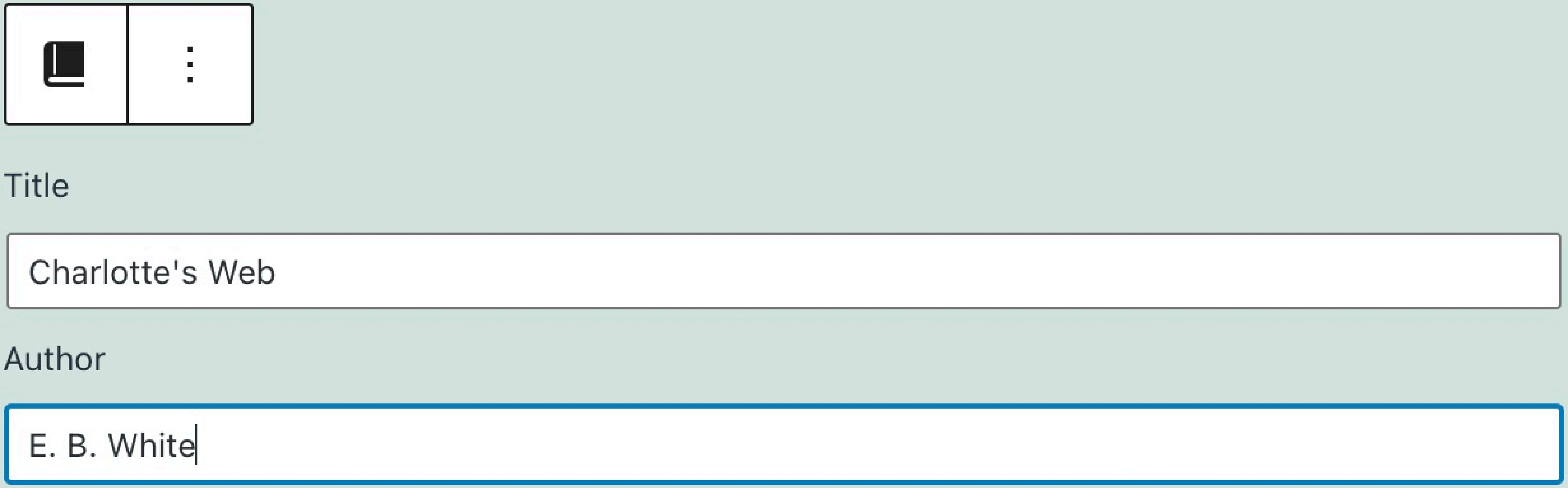
Step 2

- Remove the attribute declaration from the JS registration
- Update the `save()` JS function to return `null`

```
registerBlockType( 'my/book', {
  title: 'Book',
  description: 'A simple book block.',
  category: 'text',
  icon: 'book-alt',

  edit: ( props ) => {
    return ( BookEdit( props ) );
  },

  save: () => {
    return null;
  }

} );
```

Step 3

Nothing! The edit function is the only thing left and that can stay the same.


Title

Charlotte's Web

Author

E. B. White

My Favorite Books

Charlotte’s Web

E. B. White

In the database, take 2

```
<!-- wp:my/book {"title":"Charlotte's Web",
"author":"E. B. White"} /-->
```

One more request...

"This is wonderful!" Jordan ponders, "But... can we also add a summary of the book?"

Yes! So easy, for real this time!

Book block, v3

- Add the `summary` attribute in the PHP
- Update the PHP `render()` function
- Add a control to the JS edit interface

Step 1

Update the PHP.

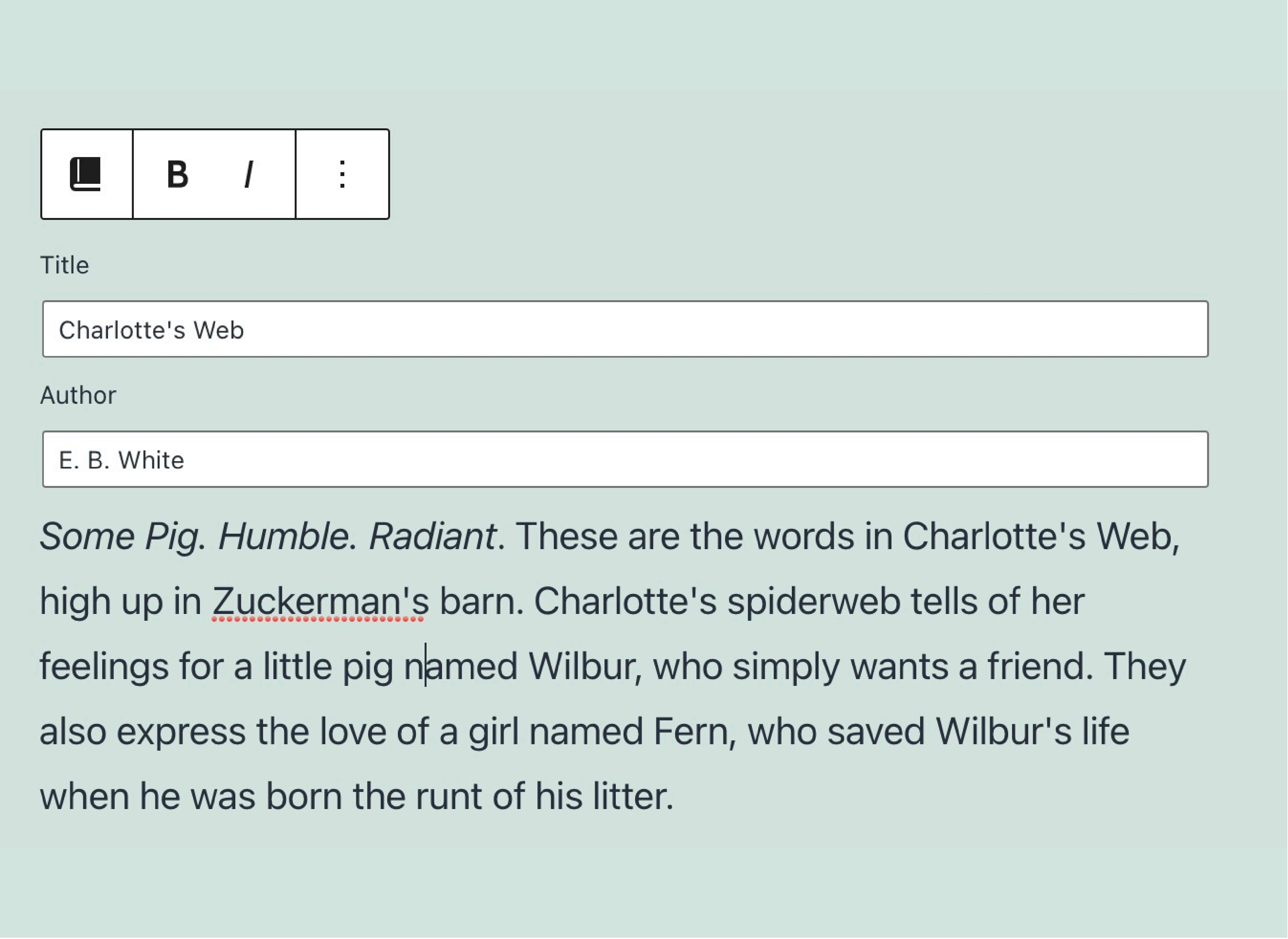
```
'attributes' => [
  'title'   => [ 'type' => 'string', 'default' => '' ],
  'author'  => [ 'type' => 'string', 'default' => '' ],
  'summary' => [ 'type' => 'string', 'default' => '' ]
]
```

```
public function render( $attributes ): string {
  $title = $attributes['title'];
  $author = $attributes['author'];
  $summary = $attributes['summary'];

  return <<<HTML
    <div class="my-book-block">
      <h3>$title</h3>
      <p>$author</p>
      <p>$summary</p>
    </div>
HTML;
}
```

Step 2

Update the JS `edit()` function.

```
const BookEdit = ( props ) => {

  // Get WP packages.
  const { RichText } = wp.blockEditor;
  const { TextControl } = wp.components;
  const { Fragment } = wp.element;

  // Get block properties.
  const { setAttributes } = props;
  const { title, author, summary } = props.attributes;

  // Set on change callbacks.
  const onChangeTitle = ( value ) => {
    setAttributes( { title: value } )
  };

  const onChangeAuthor = ( value ) => {
    setAttributes( { author: value } )
```



Title

Charlotte's Web

Author

E. B. White

Some Pig. *Humble*. *Radiant*. These are the words in Charlotte's Web, high up in Zuckerman's barn. Charlotte's spiderweb tells of her feelings for a little pig named Wilbur, who simply wants a friend. They also express the love of a girl named Fern, who saved Wilbur's life when he was born the runt of his litter.

My Favorite Books

Charlotte’s Web

E. B. White

Some Pig. *Humble*. *Radiant*. These are the words in Charlotte’s Web, high up in Zuckerman’s barn. Charlotte’s spiderweb tells of her feelings for a little pig named Wilbur, who simply wants a friend. They also express the love of a girl named Fern, who saved Wilbur’s life when he was born the runt of his litter.

Summary

- Static blocks are great for static data.
- However, if your static block markup changes, you must declare deprecations.
- Dynamic blocks can avoid this - useful if you have a static block that you know will evolve over time.

These slides

https://talks.jhalabi.com/dynamic-blocks

Example code

Static block: https://github.com/That-Dev-Girl-Sites/talks.jhalabi.com/tree/main/dynamic-blocks/examples/static

Dynamic block: https://github.com/That-Dev-Girl-Sites/talks.jhalabi.com/tree/main/dynamic-blocks/examples/dynamic

Thank you!!

- Tweet at me! @jonihalabi
- Check out my GitHub: @thatdevgirl
- Other randomness at jhalabi.com