v17 Editor experience:
A template for success

Joe Glombek

Senior Developer at Bump Digital

- 5× Umbraco MVP
- AngularJS filter cheat sheet

- @joe@umbraco community.social
- @joe.gl

Why does backoffice UX matter?

the way our last agency built

“*We don’t really like*^*Umbraco*”

— Prospective clients?

> “*I don’t get the new Umbraco backoffice*”
>
> — Me, until very recently

Property descriptions

Property descriptions

Useful to explain a property when a name won’t do it.

Markdown!

HTML!

UUI!

Property Descriptions: Read more

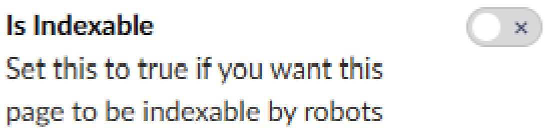
The old way:

```
Short description

---

Descriptions below a `---` were rendered behind a "Read More" link in v9-13.
```

Verses in modern Umbraco:

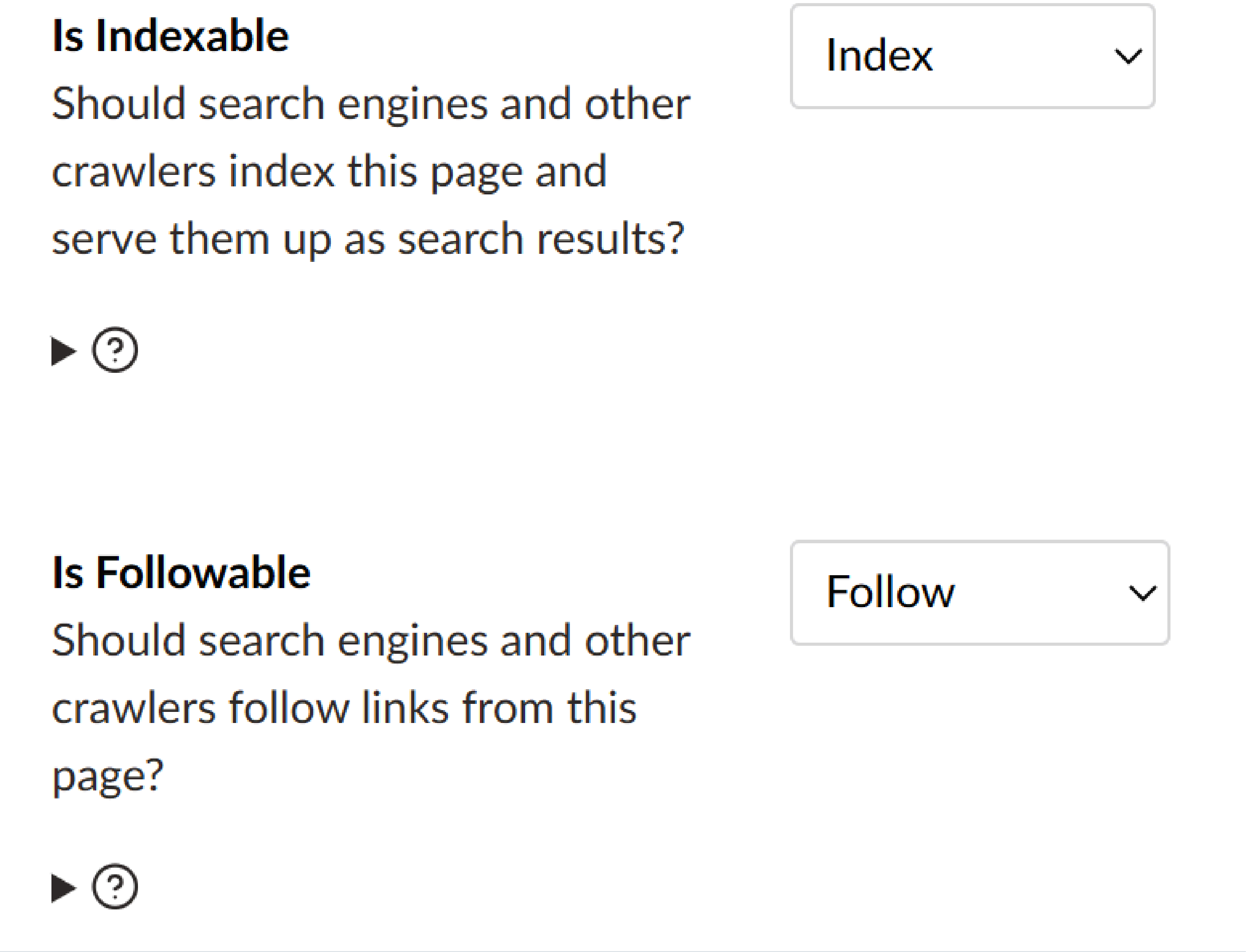
```
Short description

---

Three dashes renders a horizontal rule.

<details>
<summary>Read more</summary>

We have to use the native HTML `details` element for modern Umbraco.

</details>
```

```
Should search engines and other crawlers index this page and serve them up as search results?
<details>
<summary>
  <uui-icon name="icon-help-alt" label="More details"></uui-icon>
</summary>

This sets the *index* aspect of the [`robots` meta tag](https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Elements/meta/name/robots)

</details>
```



Is Indexable ×

Set this to true if you want this page to be indexable by robots



Is Indexable

Should search engines and other crawlers index this page and serve them up as search results?

Index ⌄

▶ ?

Is Followable

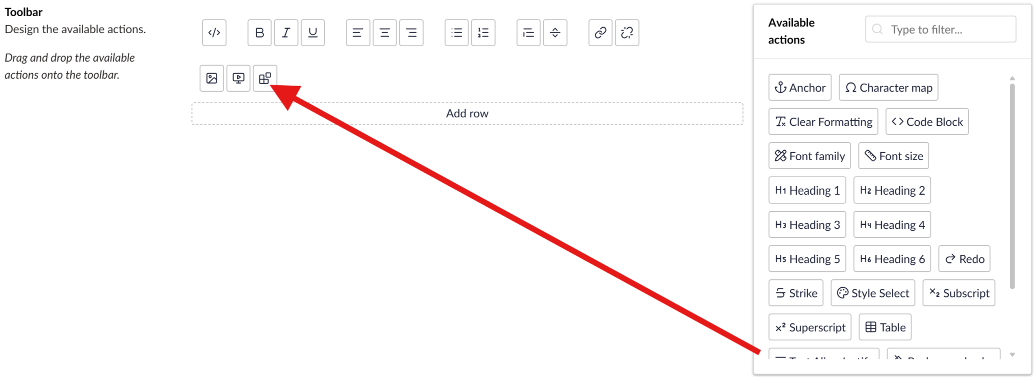
Should search engines and other crawlers follow links from this page?

Follow ⌄

▶ ?

Rich Text Editor Blocks

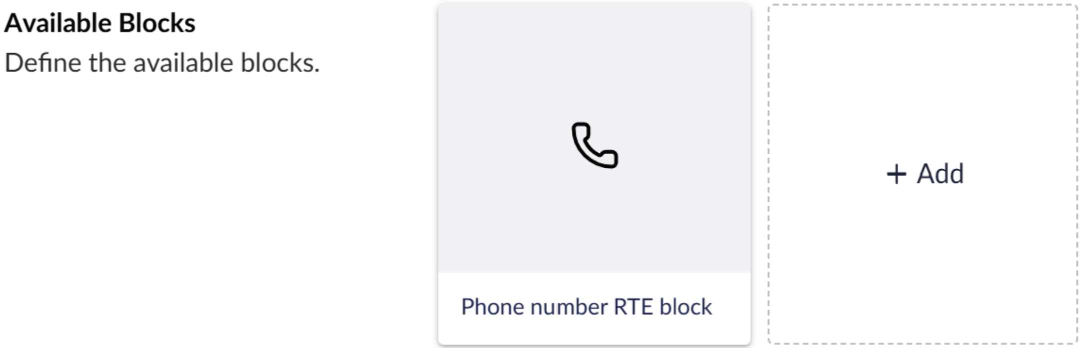
RTE Blocks Setup



Toolbar

Design the available actions.

Drag and drop the available actions onto the toolbar.



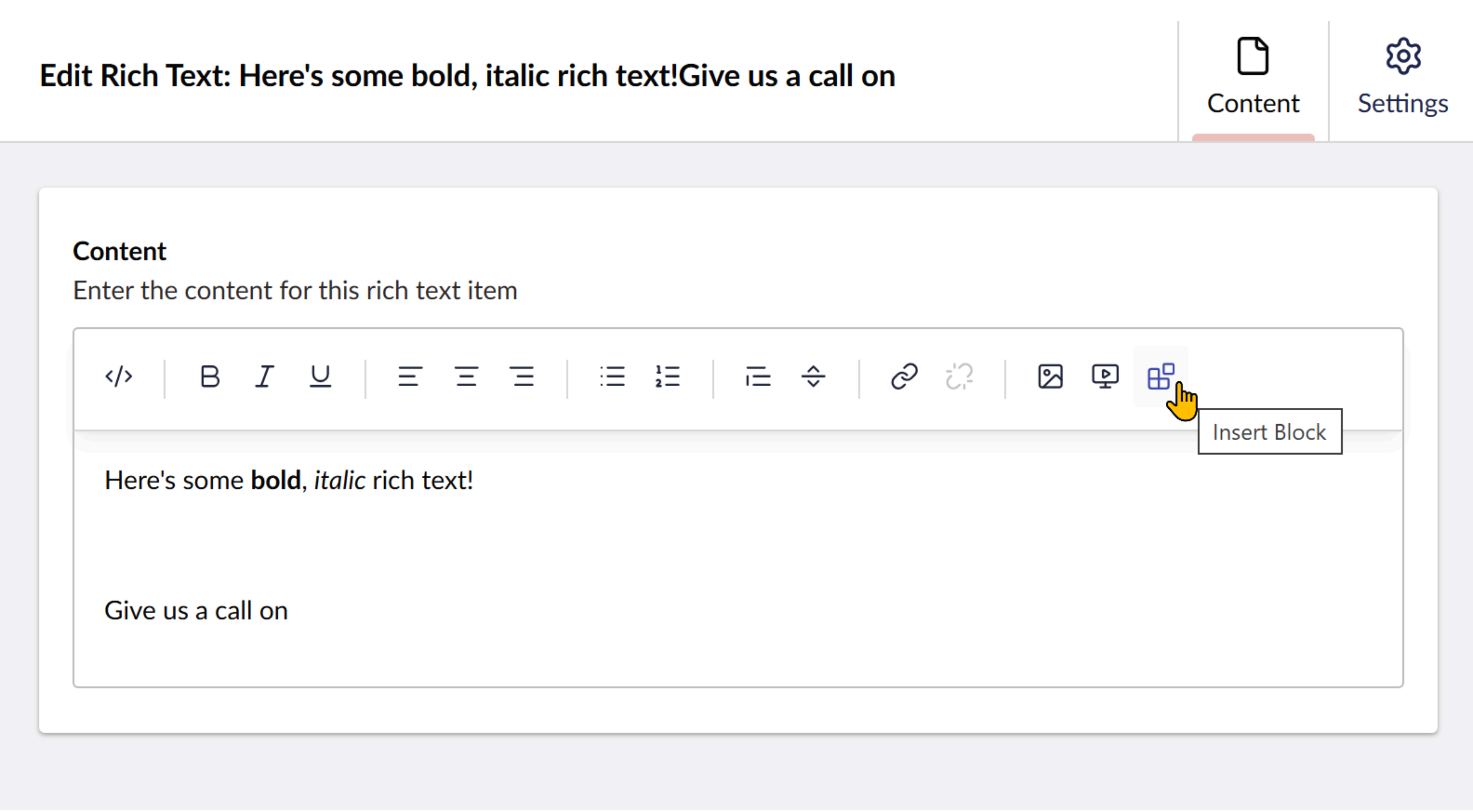
Available Blocks

Define the available blocks.

RTE Block views

@ TemplateForSuccess.Web\Views\Partials\RichText\Components\PhoneNumberRteBlock.cshtml

```
@inherits Umbraco.Cms.Web.Common.Views.UmbracoViewPage<Umbraco.Cms.Core.Models.Blocks.RichTextBlockItem<ContentModels.PhoneNumberRteBlock>>
@using ContentModels = Umbraco.Cms.Web.Common.PublishedModels;
@using System.Text.RegularExpressions
@{
  var contactPage = Model.Content.ContactUsPage as ContentModels.Contact;

  if(contactPage is null || string.IsNullOrWhiteSpace(contactPage.PhoneNumber))
  {
    return;
  }
}

<a href="tel:@contactPage.PhoneNumber.Replace(" ", "")">
  @Regex.Replace(contactPage.PhoneNumber, @"\+44\s*", "0")
</a>
```



Edit Rich Text: Here's some bold, italic rich text!Give us a call on

Content

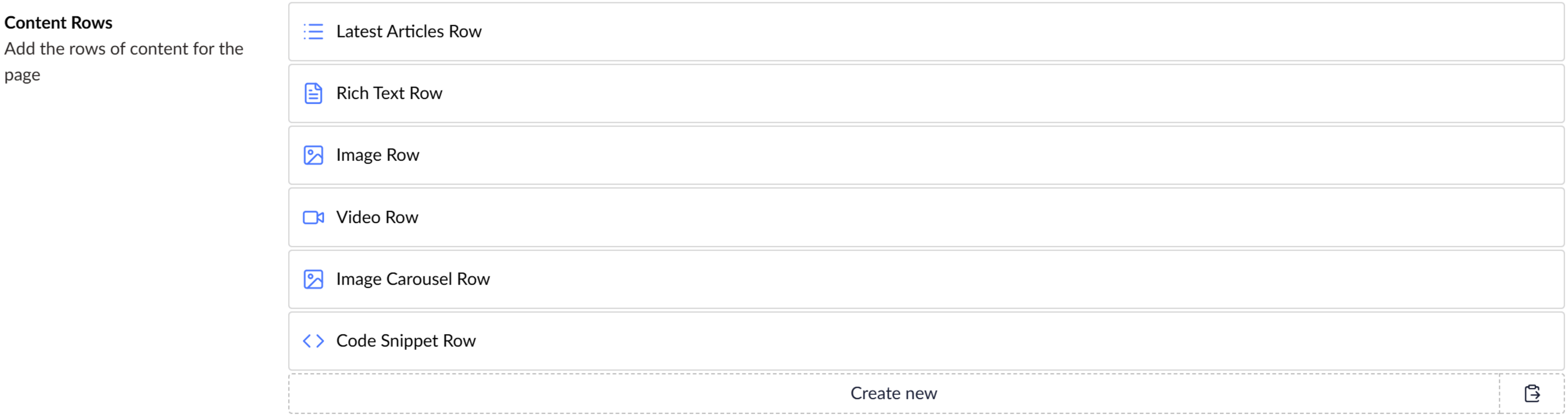
Enter the content for this rich text item

Here's some **bold**, *italic* rich text!

Give us a call on

Here's some **bold**, *italic* rich text!

Give us a call on 0123 456 7890



Content Rows

Add the rows of content for the page

- Latest Articles Row
- Rich Text Row
- Image Row
- Video Row
- Image Carousel Row
- Code Snippet Row

Block label templates

Block label templates

We used to be able to achieve this using AngularJS templates where we could do things like:

```
Call to Action: {{ page | ncNodeName }}
```

```
Text module: {{ bodyText | ncRichText | truncate:true:35 }}
```

Umbraco Flavored Markdown (UFM)

There are several syntaxes for using UFM:

Components

`{componentAlias:value}`

Expressions

`${ jsLikeSyntax }`

Filters

`| filterAlias:parameters`

appended to components *or* expressions

UFM Components

Component	Use	Example
`umbValue`	rendering the value of a property	`{umbValue: heading}`
`umbContentName`	getting the name(s) of picked content	`{umbContentName: blogCategory}`
`umbLink`	gets the title of a picked link	`{umbLink: callToAction}`
`umbLocalize`	localizing a dictionary string	`{umbLocalize: contact_us}`

- More limited than Expressions
- Only method of obtaining content names and localization keys (currently)
- Filters with parameters currently only work for components (bug)

$ UFM Expressions

- More like AngularJS templates
- Simple, safe JS-like expressions
- Access to content properties, `$settings`, `$index` (0-based)
- Native non-global JS functions

```
${ $index+1 }. Rich Text: ${ content } ${ $settings.hide == '1' ? '[HIDDEN]' : '' }
```

UFM Filters

Filter	Parameters	Use
`wordLimit`	number of words	limits to a number of words
`truncate`	length, suffix	limits to a number of characters and appends a suffix (`…`) if trunctated
`stripHtml`		removes HTML markup leaving only the text
`uppercase`		converts text to UPPER CASE
`lowercase`		converts text to lower case
`titleCase`		converts text to Title Case
`fallback`	fallback value	taking a string parameter to show if the value would otherwise be null
`bytes`		formats a number of bytes as human-readable text in KB, MB, GB, etc.

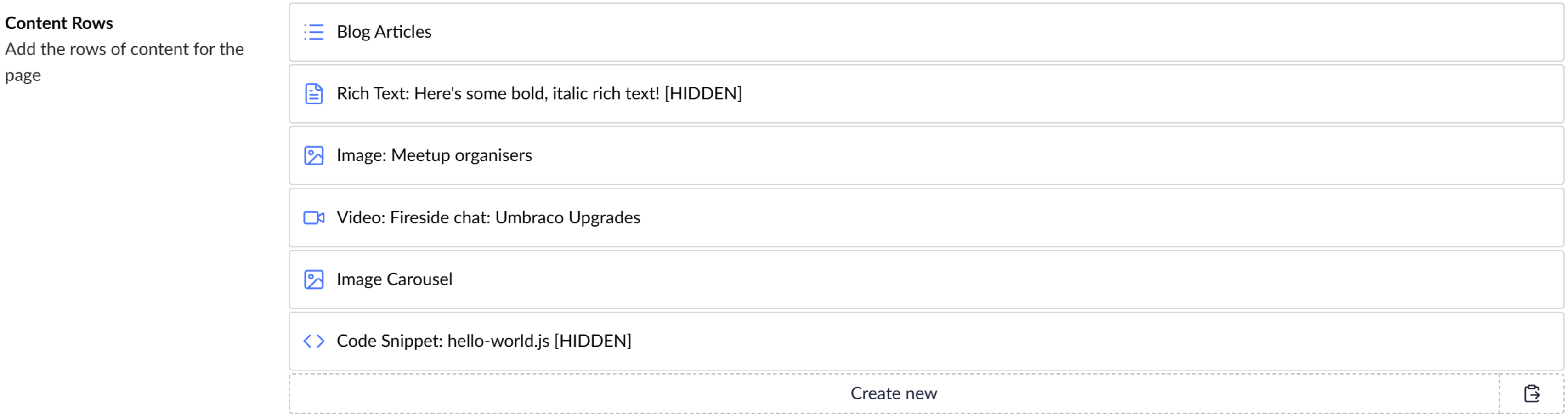
UFM Filters

The filters allow us to tidy up our example from earlier by stripping the HTML, truncating and providing a fallback.

```
${ $index+1 }. Rich Text: {umbValue: content | stripHtml | truncate:30 | fallback:[Empty] } ${ $settings.hide == '1' ? '[HIDDEN]' : '' }
```

Filters with parameters don’t currently work for expressions

```
Rich Text: ${ content | stripHtml } ${ $settings.hide == '1' ? '[HIDDEN]' : '' }
Image: ${ caption } ${ $settings.hide == '1' ? '[HIDDEN]' : '' }
Video: ${ caption != '' ? caption : videoUrl } ${ $settings.hide == '1' ? '[HIDDEN]' : '' }
Code Snippet: ${ title } ${ $settings.hide == '1' ? '[HIDDEN]' : '' }
Image Carousel ${ $settings.hide == '1' ? '[HIDDEN]' : '' }
{umbContentName:articleList} Articles ${ $settings.hide == '1' ? '[HIDDEN]' : '' }
```



Content Rows

Add the rows of content for the page

Blog Articles

Rich Text: Here's some bold, italic rich text! [HIDDEN]

Image: Meetup organisers

Video: Fireside chat: Umbraco Upgrades

Image Carousel

Code Snippet: hello-world.js [HIDDEN]

Extending UFM

- AngularJS templates were so flexible it’s hard for Umbraco to know what people were doing with it!
- Not every use case has been replicated
- Features are being added quickly when requested
- Mechanisms to extend UFM:
  - custom components
  - custom filters
    synchronous functions that are best used for basic string manipulation and cannot render HTML

Extending the Umbraco Backoffice

📦 Extending the Umbraco Backoffice

- Changed significantly with v14+
- Geared up for packages!
- Harder for one-off tweaks to sites

📦 Creating an extension

- Vite Package Setup - Umbraco Docs
- Umbraco Extension Template - Umbraco Docs
- Lotte’s Opinionated Umbraco Package Starter Template
- Vanilla JS (No, you don’t need Lit, Vite, or TypeScript to Extend the Umbraco Backoffice - Luuk Peters)
- Bump’s Umbraco Backoffice Extension Starter

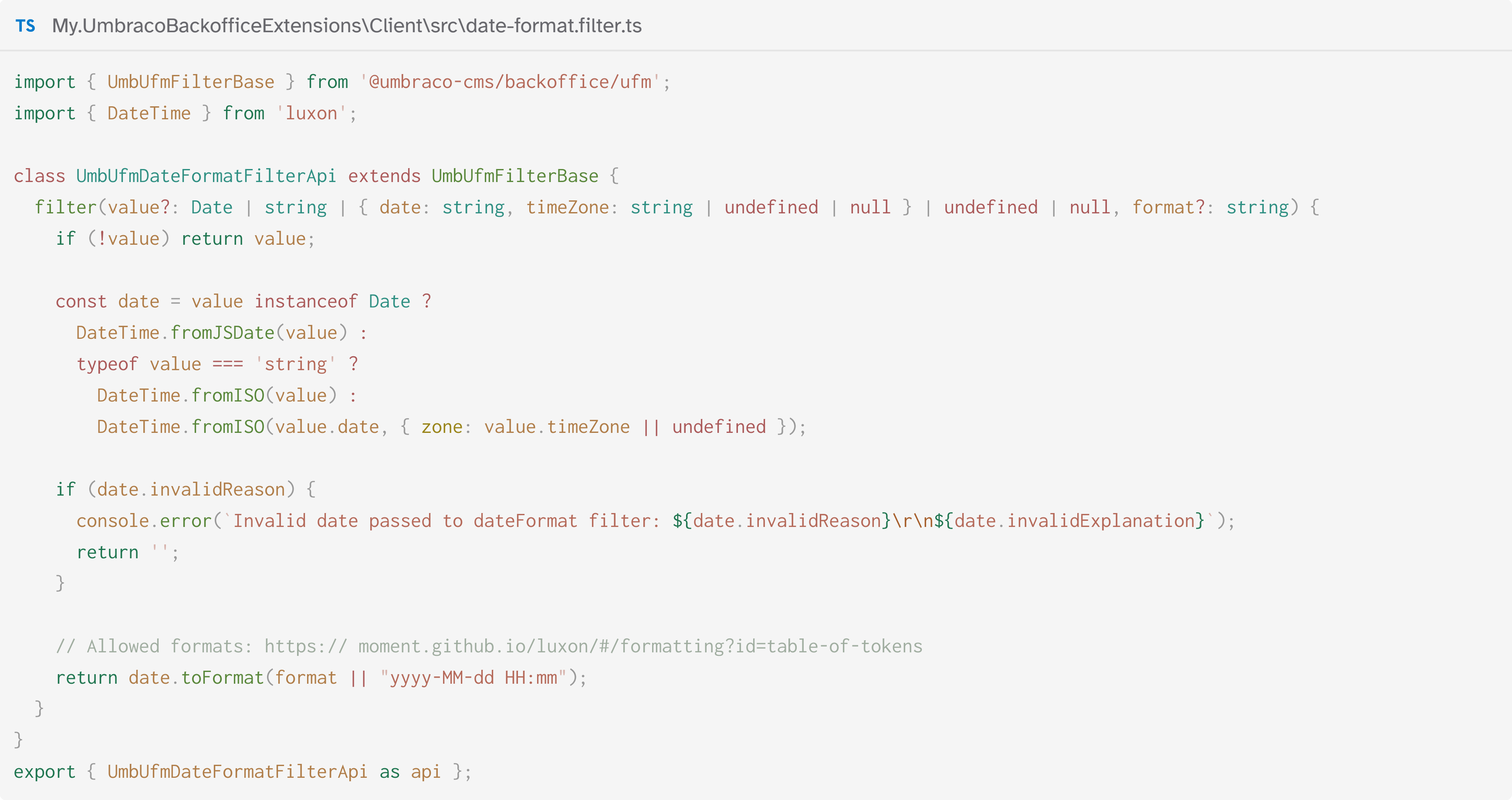
Creating a custom UFM Filter

Creating a custom UFM Filter

TS My.UmbracoBackofficeExtensions\Client\src\manifests.ts

```ts
export const manifests: Array<UmbExtensionManifest> = [
  {
    type: 'ufmFilter',
    alias: 'My.UfmFilter.DateFormat',
    name: 'Date Format UFM Filter',
    api: () => import('./date-format.filter'),
    meta: {
      alias: 'dateFormat'
    }
  }
];
```

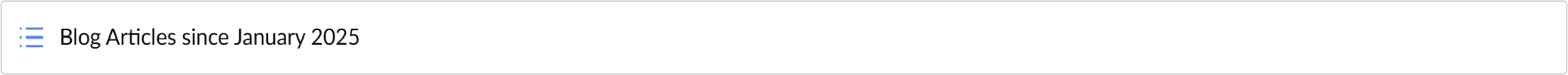
Creating a custom UFM Filter



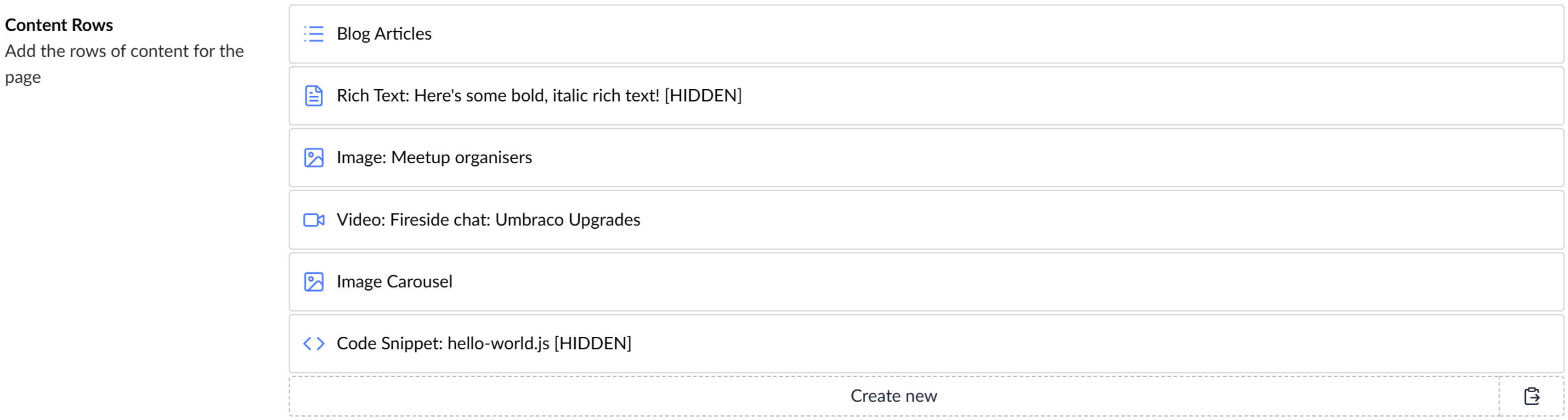
TS My.UmbracoBackofficeExtensions\Client\src\date-format.filter.ts

```typescript
import { UmbUfmFilterBase } from '@umbraco-cms/backoffice/ufm';
import { DateTime } from 'luxon';

class UmbUfmDateFormatFilterApi extends UmbUfmFilterBase {
  filter(value?: Date | string | { date: string, timeZone: string | undefined | null } | undefined | null, format?: string) {
    if (!value) return value;

    const date = value instanceof Date ?
      DateTime.fromJSDate(value) :
      typeof value === 'string' ?
        DateTime.fromISO(value) :
        DateTime.fromISO(value.date, { zone: value.timeZone || undefined });

    if (date.invalidReason) {
      console.error(`Invalid date passed to dateFormat filter: ${date.invalidReason}\r\n${date.invalidExplanation}`);
      return '';
    }

    // Allowed formats: https:// moment.github.io/luxon/#/formatting?id=table-of-tokens
    return date.toFormat(format || "yyyy-MM-dd HH:mm");
  }
}
export { UmbUfmDateFormatFilterApi as api };
```

```
{umbContentName:articleList} Articles${ dateFrom ? ' since ' : '' }{umbValue: dateFrom|dateFormat:MMMM yyyy} ${$settings.hide == '1' ? '[HIDDEN]' : ''}
```



Blog Articles since January 2025



Content Rows

Add the rows of content for the page

- Blog Articles
- Rich Text: Here's some bold, italic rich text! [HIDDEN]
- Image: Meetup organisers
- Video: Fireside chat: Umbraco Upgrades
- Image Carousel
- Code Snippet: hello-world.js [HIDDEN]

Creating a custom UFM Component

Creating a custom UFM Component

TS My.UmbracoBackofficeExtensions\Client\src\manifests.ts

```ts
export const manifests: Array<UmbExtensionManifest> = [
  {
    type: 'ufmFilter',
    alias: 'My.UfmFilter.DateFormat',
    name: 'Date Format UFM Filter',
    api: () => import('./date-format.filter'),
    meta: {
      alias: 'dateFormat'
    }
  },
  {
    type: 'ufmComponent',
    alias: 'My.UfmComponent.Tag',
    name: 'Tag UFM Component',
    api: () => import('./tag.component'),
    meta: {
      alias: 'tag'
    }
  }
];
```

Creating a custom UFM Component

TS My.UmbracoBackofficeExtensions\Client\src\tag.component.ts

```ts
//...
export class TagUfmComponentApi extends UmbUfmComponentBase {
  constructor() { ... }
  render(token: Tokens.Generic) {
    if (!token.text) return;

    const attributes = this.getAttributes(token.text);
    return `<ufm-my-tag ${attributes}></ufm-my-tag>`;
  }
  protected override getAttributes(text: string): string | null {
    if (!text) return null;

    const pipeIndex = text.indexOf('|');
    const left = text.substring(0, pipeIndex == -1 ? undefined: pipeIndex).trim();
    const filters = pipeIndex === -1 ? null : text.substring(pipeIndex + 1).trim();
    const parts = left.split(':'), alias = parts[0].trim(), display = parts[1]?.trim(), color = parts[2]?.trim(), look = parts[3]?.trim();

    return Object.entries({ alias, filters, display, color, look })
      .map(([key, value]) => (value ? `${key}="${value.trim()}"` : null))
      .join(' ');
  }
}
//...
```

Creating a custom UFM Component

TS My.UmbracoBackofficeExtensions\Client\src\tag.element.ts

```
//...
export class UmbUfmLabelValueElement extends UmbUfmElementBase  {
  constructor() {
    this.consumeContext(UMB_UFM_RENDER_CONTEXT, (context) => {
      this.observe(
        context?.value,
        (value) => {
          if (this.alias !== undefined && value !== undefined && typeof value === 'object') {
            this.value = (value as Record<string, unknown>)[this.alias];
          } else {
            this.value = value;
          }
          //TODO: manipulate the value here, or look up other values, etc
        },
        'observeValue',
      );
    });
  }
//...
```

```
import { customElement, property, state, html, css } from '@umbraco-cms/backoffice/external/lit';
import { UMB_UFM_RENDER_CONTEXT } from '@umbraco-cms/backoffice/ufm';
import { UmbLitElement } from '@umbraco-cms/backoffice/lit-element';

@customElement('ufm-my-tag')
// Doesn't have to extend UmbUfmElementBase, that class forces render to return a string rather than an HTML template,
// but does include some filter logic (which this element just ignores anyway)
export class UmbUfmLabelValueElement extends UmbLitElement  {
  @property()
  alias?: string;

  @property()
  display?: string;

  @property()
  color?: string;

  @property()
  look?: string;

  @state()
  show: Boolean;

  constructor() {
    super();

    this.show = false;

    this.consumeContext(UMB_UFM_RENDER_CONTEXT, (context) => {
      this.observe(
        context?.value,
        (value) => {
          if (this.alias !== undefined && value !== undefined && typeof value === 'object') {
            var obj = value as Record<string, unknown>;
```

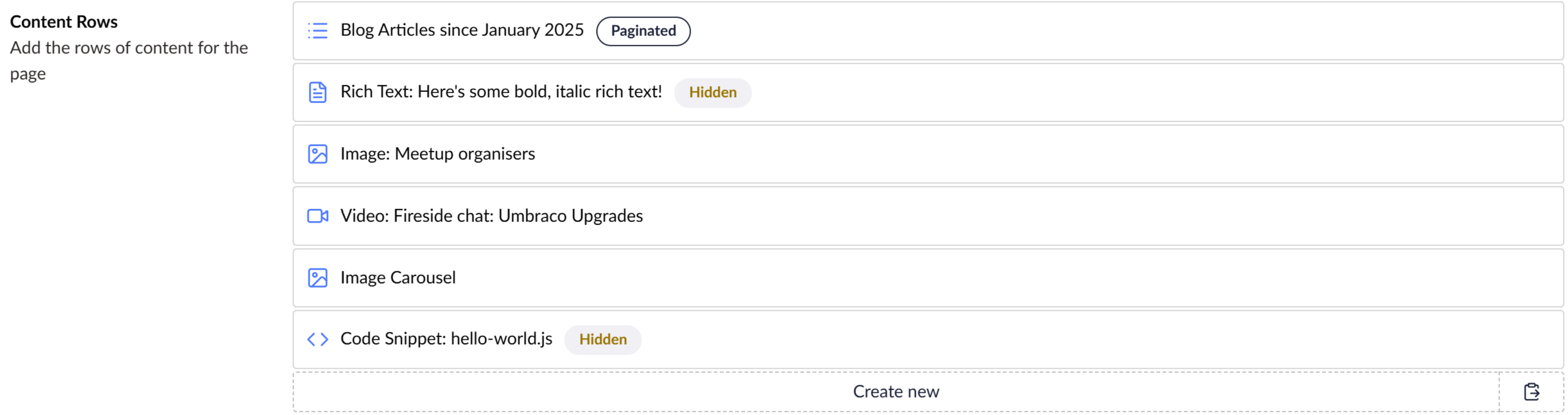
```
${ $settings.hide == '1' ? '[HIDDEN]' : '' }
```

replaced with

```
{tag: hide:Hidden:warning:secondary}
```

Also added

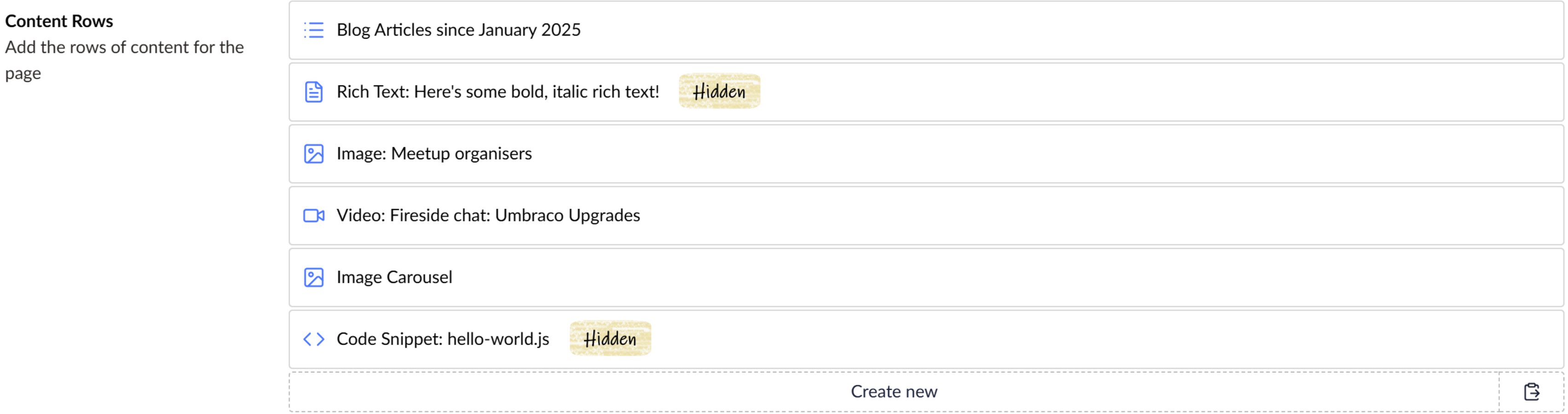
```
{tag: showPagination:Paginated:default:outline}
```



Content Rows
Add the rows of content for the page

- Blog Articles since January 2025 Paginated
- Rich Text: Here's some bold, italic rich text! Hidden
- Image: Meetup organisers
- Video: Fireside chat: Umbraco Upgrades
- Image Carousel
- Code Snippet: hello-world.js Hidden

Let’s take that "Hidden" example further...



Content Rows
Add the rows of content for the page

- Blog Articles since January 2025
- Rich Text: Here's some bold, italic rich text! Hidden
- Image: Meetup organisers
- Video: Fireside chat: Umbraco Upgrades
- Image Carousel
- Code Snippet: hello-world.js Hidden

We add the hidden logic to every block label. What if we could avoid adding anything to the label template at all?

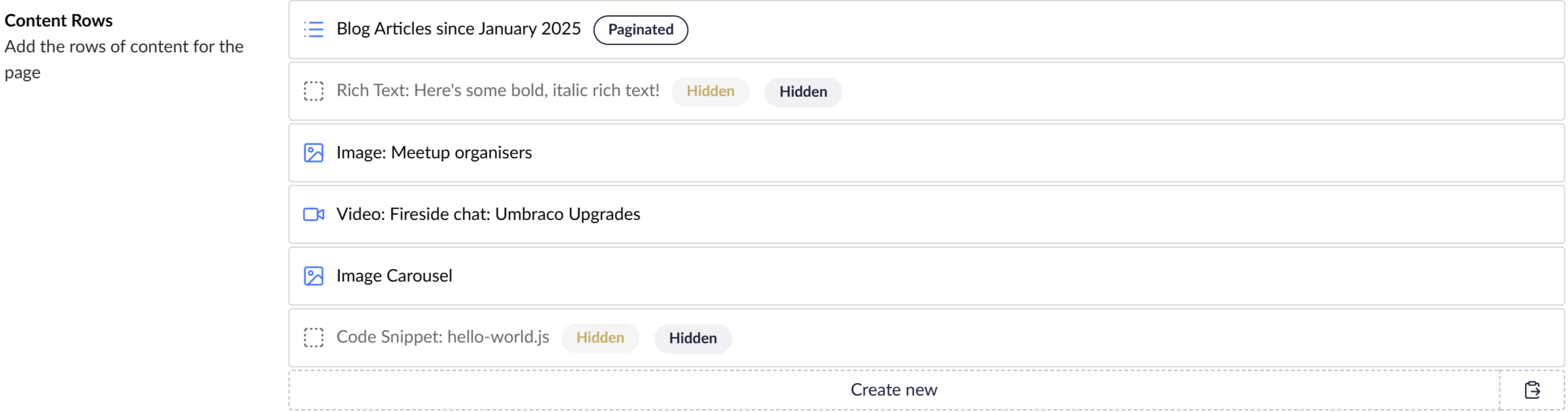
Custom block views

Custom block views

TS My.UmbracoBackofficeExtensions\Client\src\manifests.ts

```ts
export const manifests: Array<UmbExtensionManifest> = [
  // ...
  {
    type: 'ufmComponent',
    alias: 'My.UfmComponent.Tag',
    name: 'Tag UFM Component',
    api: () => import('./tag.component'),
    meta: {
      alias: 'tag'
    }
  },
  {
    type: 'blockEditorCustomView',
    alias: 'My.HiddenBlockEditorView',
    name: 'Hidden Block Editor View',
    element: () => import('./hidden-block.element'),
    forBlockEditor: 'block-list'
  }
];
```

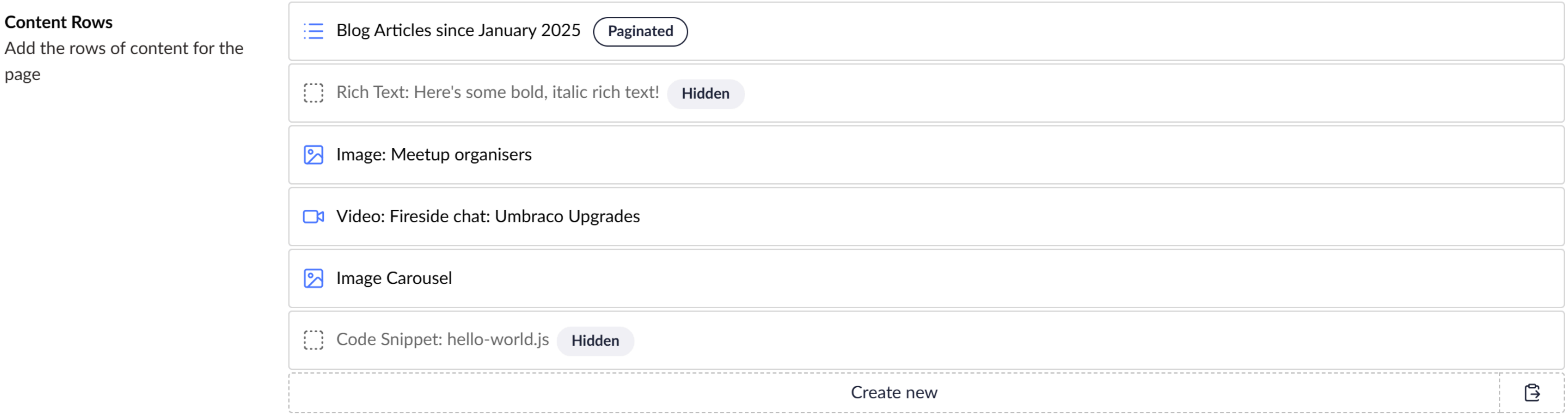
```
import { html, customElement, LitElement, property, css, when } from '@umbraco-cms/backoffice/external/lit';
import { UmbElementMixin } from '@umbraco-cms/backoffice/element-api';
import type { UmbBlockDataType } from '@umbraco-cms/backoffice/block';
import type { UmbBlockEditorCustomViewElement, UmbBlockEditorCustomViewConfiguration } from '@umbraco-cms/backoffice/block-custom-view';

@customElement('hidden-block-custom-view')
// UmbRefListBlockElement is not exposed to extend it, so we have to copy a lot of it in to replace it
export class HiddenBlockCustomView extends UmbElementMixin(LitElement) implements UmbBlockEditorCustomViewElement {

  @property({ type: String, reflect: false })
  label?: string;

  @property({ type: String, reflect: false })
  icon?: string;

  @property({ type: Number, attribute: false })
  index?: number;

  @property({ type: Boolean, reflect: true })
  unpublished?: boolean;

  @property({ attribute: false })
  content?: UmbBlockDataType;

  @property({ attribute: false })
  settings?: UmbBlockDataType;

  @property({ attribute: false })
  config?: UmbBlockEditorCustomViewConfiguration;

  override render() {
    const blockValue = { ...this.content, $settings: this.settings, $index: this.index };
    return html`
      <uui-ref-node-standalone
```



Content Rows

Add the rows of content for the page

- Blog Articles since January 2025 Paginated
- Rich Text: Here's some bold, italic rich text! Hidden Hidden
- Image: Meetup organisers
- Video: Fireside chat: Umbraco Upgrades
- Image Carousel
- Code Snippet: hello-world.js Hidden Hidden



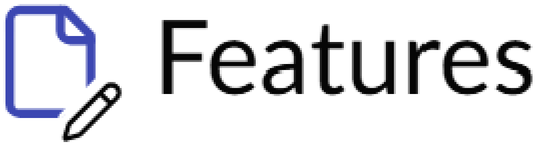
Content Rows
Add the rows of content for the page

- Blog Articles since January 2025 Paginated
- Rich Text: Here's some bold, italic rich text! Hidden
- Image: Meetup organisers
- Video: Fireside chat: Umbraco Upgrades
- Image Carousel
- Code Snippet: hello-world.js Hidden

Custom entity signs

Custom entity signs

A brand new extension point! Allows for custom signs on entities, just like the "pending changes" sign



Features

Custom entity signs

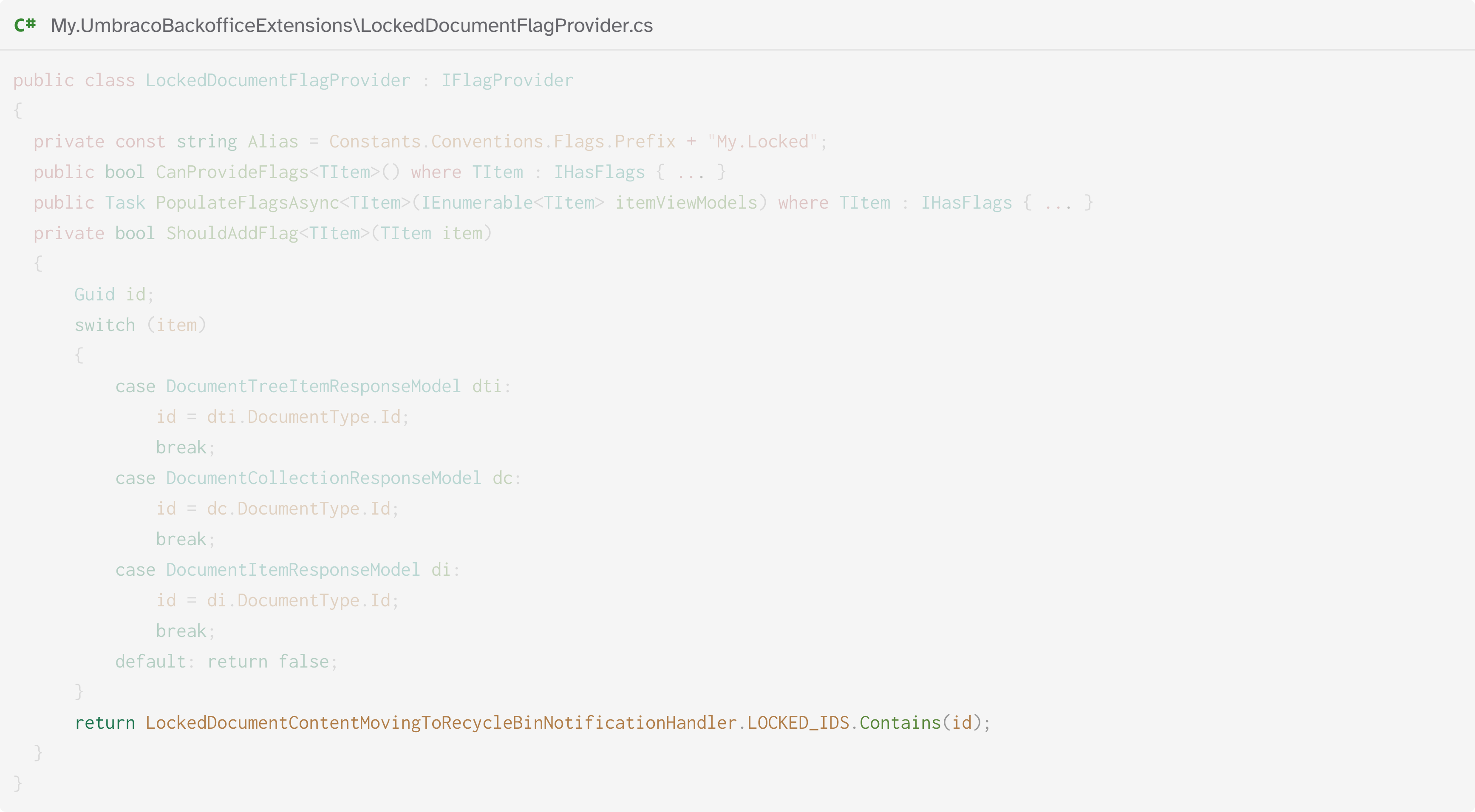
C# My.UmbracoBackofficeExtensions\Notifications\LockedDocumentContentMovingToRecycleBinNotificationHandler.cs

```csharp
/// A common use case: prevent certain document types being deleted
public class LockedDocumentContentMovingToRecycleBinNotificationHandler : INotificationHandler<ContentMovingToRecycleBinNotification>
{
    public static string[] LOCKED_ALIASES = ["home", "error"];
    public static Guid[] LOCKED_IDS = [
      Guid.Parse("a95360e8-ff04-40b1-8f46-7aa4b5983096"),
      Guid.Parse("9db112c5-c2ea-441d-8bd4-6daf522aa2b6")
    ];
    public void Handle(ContentMovingToRecycleBinNotification notification)
    {
        foreach (var item in notification.MoveInfoCollection)
        {
            if (Array.Exists(LOCKED_ALIASES, alias => alias.Equals(item.Entity.ContentType.Alias, StringComparison.OrdinalIgnoreCase)))
            {
                notification.CancelOperation(new EventMessage(
                  $"{item.Entity.Name} cannot be trashed",
                  $"The content item '{item.Entity.Name}' is of type '{item.Entity.ContentType.Name}' which cannot be trashed.",
                  EventMessageType.Error));
            }
        }
    }
}
```

Custom entity signs

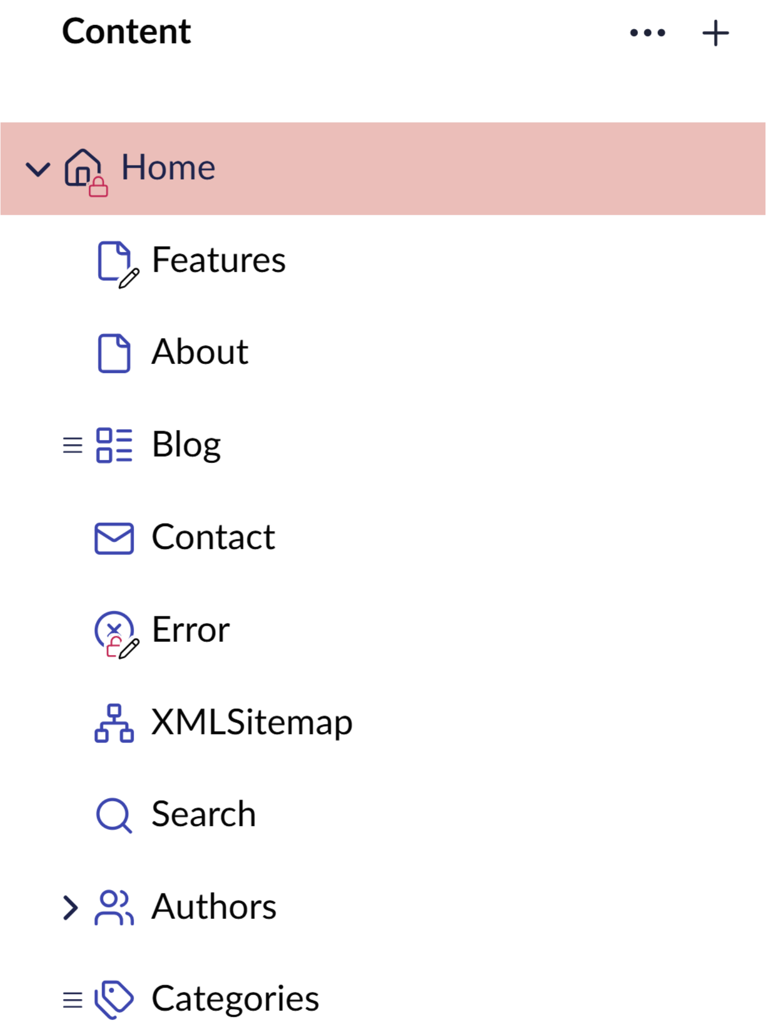
TS My.UmbracoBackofficeExtensions\Client\src\manifests.ts

```
import { UMB_DOCUMENT_ENTITY_TYPE } from '@umbraco-cms/backoffice/document';

export const manifests: Array<UmbExtensionManifest> = [
  // ...
    name: 'Hidden Block Editor View',
    element: () => import('./hidden-block.element'),
    forBlockEditor: 'block-list'
  },
  {
    type: 'entitySign',
    kind: 'icon',
    alias: 'Umb.EntitySign.Document.My.Locked',
    name: 'Is Locked Document Entity Sign',
    forEntityTypes: [UMB_DOCUMENT_ENTITY_TYPE],
    forEntityFlags: ['Umb.My.Locked'],
    weight: -1000,
    meta: {
      iconName: 'icon-lock',
      label: 'Locked',
      iconColorAlias: 'red',
    }
  }
];
```



C# My.UmbracoBackofficeExtensions\LockedDocumentFlagProvider.cs

```csharp
public class LockedDocumentFlagProvider : IFlagProvider
{
  private const string Alias = Constants.Conventions.Flags.Prefix + "My.Locked";
  public bool CanProvideFlags<TItem>() where TItem : IHasFlags { ... }
  public Task PopulateFlagsAsync<TItem>(IEnumerable<TItem> itemViewModels) where TItem : IHasFlags { ... }
  private bool ShouldAddFlag<TItem>(TItem item)
  {
      Guid id;
      switch (item)
      {
          case DocumentTreeItemResponseModel dti:
              id = dti.DocumentType.Id;
              break;
          case DocumentCollectionResponseModel dc:
              id = dc.DocumentType.Id;
              break;
          case DocumentItemResponseModel di:
              id = di.DocumentType.Id;
              break;
          default: return false;
      }
      return LockedDocumentContentMovingToRecycleBinNotificationHandler.LOCKED_IDS.Contains(id);
  }
}
```



Content
•••
+
Home
Features
About
Blog
Contact
Error
XMLSitemap
Search
Authors
Categories

Your new customisation options...

Property descriptions

RTE Blocks

UFM Components + custom

UFM Expressions

UFM Filters + custom

Extending the Backoffice

Custom block views

Custom entity signs

…as well as the old ones…

Icons

Property validation (regex!)

Avoid repeating content

Semantic content modelling

List views

User testing & feedback loops

Thank you

@joe@umbraco community.social

@joe.gl

linkedin.com/in/glombek

joe.gl/ombek

joe.gl/ombek/links/template-for-success