| **Section Metadata** | |
| --- | --- |
| style | content |

A computer screen with a pencil and a window

AI-generated content may be incorrect.

**Creating Blocks Instrumented for use with the Universal Editor**

Learn how to create blocks instrumented for the Universal Editor when using AEM authoring as your content source by adding components, loading component definitions in the Universal Editor, publishing pages, implementing block decoration and styles, bringing the changes to production, and verifying them.

**Prerequisites**

To create your blocks, you will need some existing knowledge of AEM authoring with Edge Delivery Services projects as well as the Universal Editor. You should also already have access to Edge Delivery Services and be familiar with its basics including:

* You have access to an [AEM Cloud Service sandbox.](https://experienceleague.adobe.com/en/docs/experience-manager-cloud-service/content/implementing/using-cloud-manager/programs/introduction-sandbox-programs)
* You have completed the [Getting Started – Universal Editor Developer Tutorial.](https://www.aem.live/developer/ue-tutorial)

**Adding a New Block to Your Project**

Let’s build a block to render a memorable quote on your page.

To simplify this example, all changes are made to the main branch of the project repository. Of course for your actual project, you should follow [development best practices](https://www.aem.live/docs/dev-collab-and-good-practices) by developing on a different branch and reviewing all changes via pull request before merging to main.

Adobe recommends that you develop blocks in a three-phased approach:

1. Create the definition and model for the block, review it, and bring it to production.
2. Create content with the new block.
3. Implement the decoration and styles for the new block.

The following quote block example follows this approach.

**Create Block Definition and Model**

1. Clone the GitHub project locally that you created in the [Getting Started – Universal Editor Developer Tutorial](https://www.aem.live/developer/ue-tutorial) and open it in an editor of your choice.
   * Microsoft Code is used in this document for illustrative purposes.

A screenshot of a computer

AI-generated content may be incorrect.

1. Edit the component-definition.json [file at the root of the project](https://experienceleague.adobe.com/en/docs/experience-manager-cloud-service/content/implementing/developing/universal-editor/component-definition) and add the following definition for your new quote block and save the file.

{

"title": "Quote",

"id": "quote",

"plugins": {

"xwalk": {

"page": {

"resourceType": "core/franklin/components/block/v1/block",

"template": {

"name": "Quote",

"model": "quote",

"quote": "<p>Think, McFly! Think!</p>",

"author": "Biff Tannen"

}

}

}

}

}

1. Edit the component-models.json file at the root of the project and add the following [model definition](https://experienceleague.adobe.com/en/docs/experience-manager-cloud-service/content/implementing/developing/universal-editor/field-types#model-structure) for your new quote block and save the file.
   * Please see the document [Content Modeling for AEM authoring as your content source](https://www.aem.live/developer/component-model-definitions) for more information about what is important to consider when creating content models.

{

"id": "quote",

"fields": [

{

"component": "richtext",

"name": "quote",

"value": "",

"label": "Quote",

"valueType": "string"

},

{

"component": "text",

"valueType": "string",

"name": "author",

"label": "Author",

"value": ""

}

]

}

1. Edit the component-filters.json [file at the root of the project](https://experienceleague.adobe.com/en/docs/experience-manager-cloud-service/content/implementing/developing/universal-editor/filtering) and add the quote block to the filter definition to allow the block to be added to any section and save the file.

{

"id": "section",

"components": [

"text",

"image",

"button",

"title",

"hero",

"cards",

"columns",

"quote"

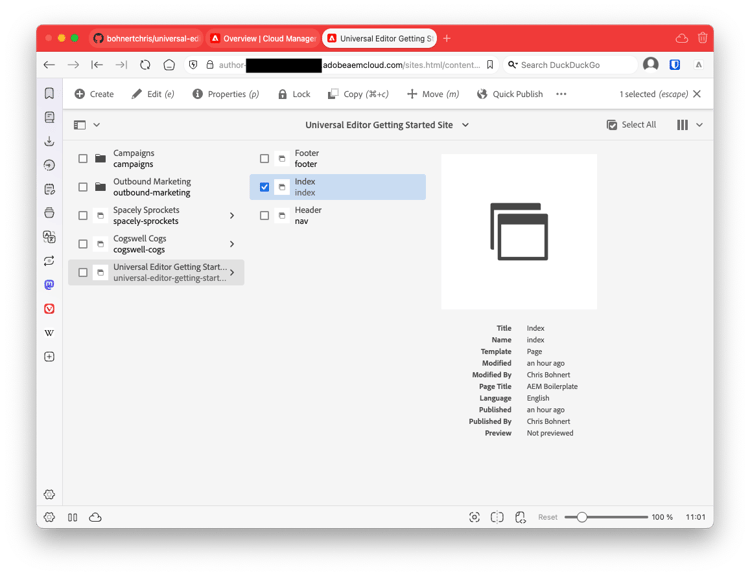
]

}

1. Using git, commit these changes to your main branch.
   * Committing to main is for illustrative purposes only. [Follow best practices](https://www.aem.live/docs/dev-collab-and-good-practices) and use a pull request for actual project work.

**Create content with the block**

Now that your basic quote block is defined and committed to the sample project, you can add a quote block to an existing page.

1. In a browser, sign into AEM as a Cloud Service. [Using the Sites console](https://experienceleague.adobe.com/en/docs/experience-manager-cloud-service/content/sites/authoring/basic-handling), navigate to the site that you created in the [Getting Started – Universal Editor Developer Tutorial](https://www.aem.live/developer/ue-tutorial) and select a page.
   * In this case, index is used for illustrative purposes.  
     
2. Tap or click **Edit** in the toolbar of the console and the Universal Editor opens.
   * In order to load the page, you may need to tap or click **Sign in with Adobe** to authenticate to AEM in the Universal Editor.
3. In the Universal Editor, select a section. In the properties panel, tap or click the **Add** icon and then select your new **Quote** block from the menu.
   * The **Add** icon is a plus symbol.
   * You know that you have selected a section if the blue outline of the selected object has a tab labeled **Section**.
   * In this example, tapping or clicking slightly above the **Lorem Ipsum** heading selects a section containing the heading and lorem ipsum text.

A screenshot of a computer

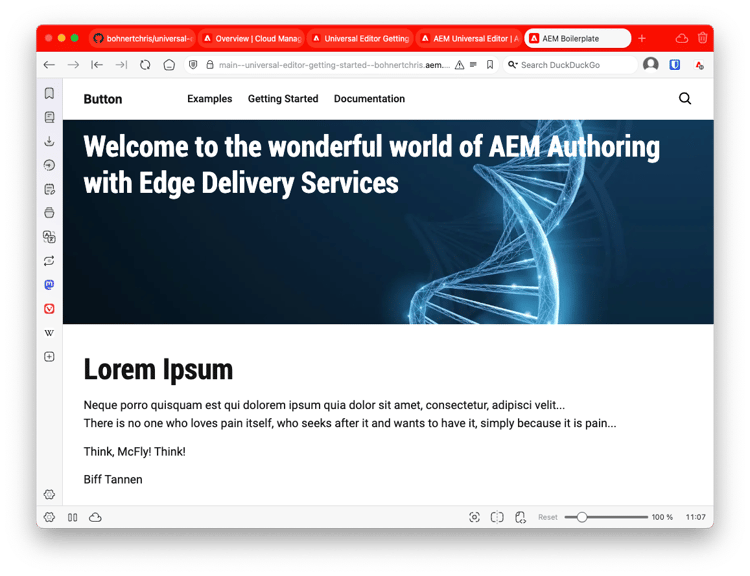
AI-generated content may be incorrect.

1. The page is reloaded and the quote block is added to the bottom of the selected section with the default content specified in the component-definitions.json file.
   * The quote block can be selected and edited as any other block either in-place or in the properties panel.
   * Styling will be applied in a further step.

A screenshot of a computer

AI-generated content may be incorrect.

1. Once you are satisfied with the content of your quote, you can publish the page by tapping or clicking the **Publish** button in the toolbar of the Universal Editor.
2. Verify that the content was published by navigating to the published page. The link will be similar to https://<branch>--<repo>--<owner>.aem.page



**Style the block**

Now that you have a working quote block you can apply styling to it.

1. Return to the editor for your project.
2. Create a **quote** folder under the blocks folder.

A screenshot of a computer

AI-generated content may be incorrect.

1. In the new quote folder, add a quote.js file to implement block decoration by adding the following JavaScript and save the file.

export default function decorate(block) {

const [quoteWrapper] = block.children;

const blockquote = document.createElement('blockquote');

blockquote.textContent = quoteWrapper.textContent.trim();

quoteWrapper.replaceChildren(blockquote);

}

1. In the quote folder, add a quote.css file to define the styling for the block by adding the following CSS code and save the file.

.block.quote {

background-color: #ccc;

padding: 0 0 24px;

display: flex;

flex-direction: column;

margin: 1rem 0;

}

.block.quote blockquote {

margin: 16px;

text-indent: 0;

}

.block.quote > div:last-child > div {

margin: 0 16px;

font-size: small;

font-style: italic;

position: relative;

}

.block.quote > div:last-child > div::after {

content: "";

display: block;

position: absolute;

left: 0;

bottom: -8px;

height: 5px;

width: 30px;

background-color: darkgray;

}.block.quote {

background-color: #ccc;

padding: 0 0 24px;

display: flex;

flex-direction: column;

margin: 1rem 0;

}

.block.quote blockquote {

margin: 16px;

text-indent: 0;

}

.block.quote > div:last-child > div {

margin: 0 16px;

font-size: small;

font-style: italic;

position: relative;

}

.block.quote > div:last-child > div::after {

content: "";

display: block;

position: absolute;

left: 0;

bottom: -8px;

height: 5px;

width: 30px;

background-color: darkgray;

}

1. Using git, commit these changes to your main branch.
   * Committing to main is for illustrative purposes only. [Follow best practices](https://www.aem.live/docs/dev-collab-and-good-practices) and use a pull request for actual project work.
2. Return to your browser tab of the Universal Editor where you were editing the page of your project and reload the page to view your styled block.
3. See the now styled quote block on the page.

A screenshot of a computer

AI-generated content may be incorrect.

1. Verify that the changes were pushed to production by navigating to the published page. The link will be similar to https://<branch>--<repo>--<owner>.aem.page

A screenshot of a computer

AI-generated content may be incorrect.

Congratulations! You now have a fully working and styled quote block. You can use this example as a basis for designing your own project-specific blocks.

**Block options**

If you need a block to look or behave slightly differently based on certain circumstances (but not different enough to become a new block in itself), you can let authors choose from [block options.](https://www.aem.live/developer/markup-sections-blocks#block-options)

By adding a classes property to the block, the property is rendered in the table header for simple blocks, or as value list for items in a container block.

{

"id": "quote",

"fields": [

{

"component": "richtext",

"name": "quote",

"value": "",

"label": "Quote",

"valueType": "string"

},

{

"component": "text",

"valueType": "string",

"name": "author",

"label": "Author",

"value": ""

},

{

"component": "select",

"name": "classes",

"value": "",

"label": "Background Color",

"description": "The quote background color",

"valueType": "string",

"options": [

{

"name": "Red",

"value": "bg-red"

},

{

"name": "Green",

"value": "bg-green"

},

{

"name": "Blue",

"value": "bg-blue"

}

]

}

]

}

**Using other working branches**

This guide had you commit directly to the main branch for simplicity’s sake. For experimentation in a sample repository, this is usually not an issue. For actual project work, you should [follow development best practices](https://www.aem.live/docs/dev-collab-and-good-practices) by developing on a different branch and reviewing all changes via pull request before merging to main.

When you are not developing on the main branch, you can append ?ref=<branch> in the Universal Editor location bar to load the page from your branch. <branch> is the branch name as it would be used for your project’s preview or live URLs, e.g. https://<branch>--<repo>--<owner>.aem.page.

**Blocks for AEM authoring and document-based authoring**

On certain projects, you may want to support both AEM authoring as your content source using the Universal Editor as well as document-based authoring. To minimize development time and ensure the same site experience, you can create one set of blocks to support both use cases.

To do this, you must use the same content modeling approach for both your AEM authoring setup as well as your document-based authoring setup.

**Approach**

In AEM authoring, you declare a model and provide naming conventions. Data is then rendered in table-like block structures using Edge Delivery in the same way as if the table would have been created manually using document-based authoring.

To achieve this, certain assumptions are made such as for a simple block like a teaser that all properties and groups of properties are rendered in 1…n rows with a single column each. For blocks that have 1…n items (such as carousel and cards) the items are appended after these rows with one row each and a column for each property/group of properties.

If you follow the same approach for document-based authoring you can reuse your AEM authoring blocks.

**Example**

The following example of a teaser block follows the recommended approach and could be used between document-based and AEM authoring.

With this data:

{

"name": "teaser",

"model": "teaser",

"image": "/content/dam/teaser-background.png",

"imageAlt": "A group of people sitting on a stage",

"teaserText\_subtitle": "Adobe Experience Cloud"

"teaserText\_title": "Meet the Experts"

"teaserText\_titleType": "h2"

"teaserText\_description": "<p>Join us in this ask me everything session...</p>"

"teaserText\_cta1": "https://link.to/more-details",

"teaserText\_cta1Text": "More Details"

"teaserText\_cta2": "https://link.to/sign-up",

"teaserText\_cta2Text": "RSVP",

"teaserText\_cta2Type": "primary"

}

You get this markup:

<div class="teaser">

<div>

<div>

<picture>

<img src="/content/dam/teaser-background.png" alt="A group of people sitting on a stage">

</picture>

</div>

</div>

<div>

<div>

<p>Adobe Experience Cloud</p>

<h2>Meet the Experts</h2>

<p>Join us in this ask me everything session ...</p>

<p><a href="https://link.to/more-details">More Details</a></p>

<p><strong><a href="https://link.to/sign-up">RSVP</a></strong></p>

</div>

</div>

</div>

And it will be turned into this table representation:

+-------------------------------------------------+

| Teaser |

+=================================================+

| ![A group of people sitting on a stage][image0] |

+-------------------------------------------------+

| Adobe Experience Cloud |

| ## Meet the Experts |

| Join us in this ask me everything session ... |

| [More Details](https://link.to/more-details) |

| [RSVP](https://link.to/sign-up) |

+-------------------------------------------------+

**Next steps**

Now that you know how to create blocks, it is essential to understand how to model content in a semantic way to achieve a lean developer experience.

| **Pagination (contained)** | |
| --- | --- |
| Previous  [**Getting Started – Universal Editor Developer Tutorial**](https://www.aem.live/developer/ue-tutorial) | Up Next  [**Content Modeling**](https://www.aem.live/developer/component-model-definitions) |

| **Metadata** | |
| --- | --- |
| Title | Creating Blocks Instrumented for use with the Universal Editor |
| Image | A computer screen with a pencil and a window  AI-generated content may be incorrect. |
| Description | Learn how to create blocks instrumented for the Universal Editor when using AEM authoring as your content source by adding components, loading component definitions in ... |
| Template | guides |
| Category | Build |