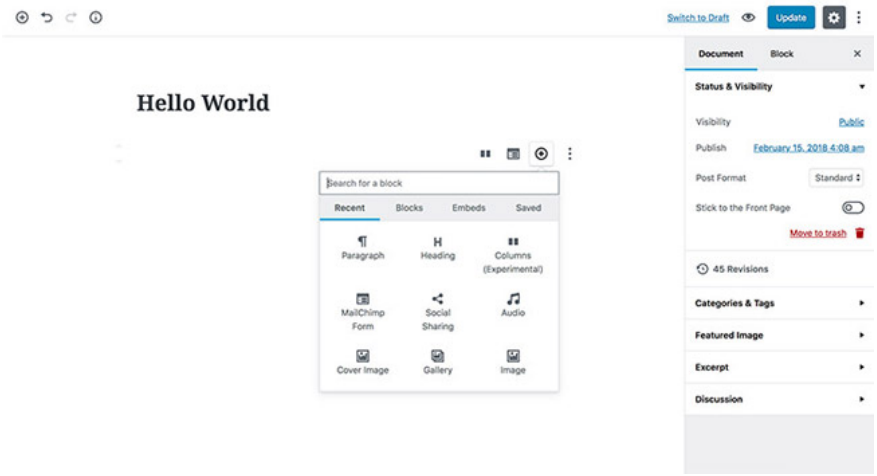
Learn about the Gutenberg Editor of WordPress

Gutenberg is a new editor for WordPress, promising to completely replace the current editor TinyMCE.

This is an ambitious project, which can change WordPress in many ways and will affect both users and developers, particularly those who depend on the editor to work on WordPress. This is Gutenberg's basic interface.



Gutenberg Editor with sidebar on the right is opened and a blocking option is displayed. Obviously, it is inspired by the user interface of Medium editor.

1. Gutenberg also introduced a new model in WordPress called "Block". "Block" is an abstract term used to describe markup units made up of the content or layout of a website. This idea combines the concepts in today's WordPress we achieve with short code, You will need this tag at the top of every generated HTML document. This ensures a browser knows that it is reading HTML. Tags Although this is not really an HTML tag, users should still know about it.
2. This is another card that tells the browser that it is reading HTML. Why do we need both tags and? It is really good to know but it is better not to leave it. And at the end of the document, add the tag.
3. For basic pages, the tag will contain the title.
4. This card specifies the title of the page. All you need to do is place the title in the tag and close it, this example includes the head tag: My Website This is the name displayed as a tab title when it is opened in a browser. Title tab
5. Like title tags, metadata is placed in the title area of ??the page (this metadata, unlike metadata from mobile devices). Metadata is primarily used by search engines and is information on your site. There are several different meta fields, but this is one of the most commonly used meta types: Description - Basic description of the page. Keywords - Select keywords that apply to the page. author - Author of the page. viewport - A card ensures that the page can be viewed on all devices. Here is an example that can be applied to this page: The "viewport" tag will always have 'width = device-width, initial-scale = 1.0' so that

the page content can display well on mobile devices and desktop computers. .
6. Basically, everything except the title is placed in the body of the web page and placed in the body tag. Everything you want to display on your page.

Set up the project

Knowing that Gutenberg is built on React, some developers are concerned that this is a huge barrier for high-end developers who want to develop Gutenberg.

Setting up React.js can be time consuming and confusing if you are a beginner. At least you will need JSX transformer, Babel. Depending on the code you have, you may need some Babel plugins and a Bundler like Webpack, Rollup or Parcel.

Fortunately, some people in the WordPress community have started and are trying to make the Gutenberg development as easy as possible for everyone to follow. Today, we have a tool to create a Gutenberg template so we can start writing code immediately instead of using existing tools and configurations.

Create Guten Block

Creating Guten Block is an initial project of Ahmad Awais. That's the zero configuration tool set (# 0CJS) that lets you develop Gutenberg blocks with some modern stacks including React, Webpack, ESNext, Babel, ESLint and Sass.

Using ES5 (ECMAScript 5)

Using all of these tools may seem too much to create a simple 'hello-world' block. If you want to keep your stack neat, you can actually develop a Gutenberg block using a familiar ECMAScript 5. If you have installed WP-CLI 1.5.0 on your computer, you can simply run:

```
wp scaffold block [--title =] [--dashicon =] [--category =] [--theme] [--plugin =
```

to create Gutenberg's template for the plugin or theme. This approach is more reasonable, especially for plugins and themes that you developed before Gutenberg.

Instead of creating a new plugin to contain Gutenberg blocks, you might want to integrate blocks into existing plugins or themes. And to make this tutorial easy for everyone, we will use ECMAScript 5 with the WP-CLI.

Register a New Block

Gutenberg is currently developed as a plugin and will be merged into WordPress 5.0 whenever the development team feels it has been completed. So during this time, you will need to install it from the Plugins page in wp-admin. Once you have installed and activated it, run the following command in Terminal or Command Prompt if you are on a Windows machine.

```
  wp scaffold block series --title = "HTML5 Series" --theme
```

This command will create a new theme Block that is currently active. The block will include the following files:

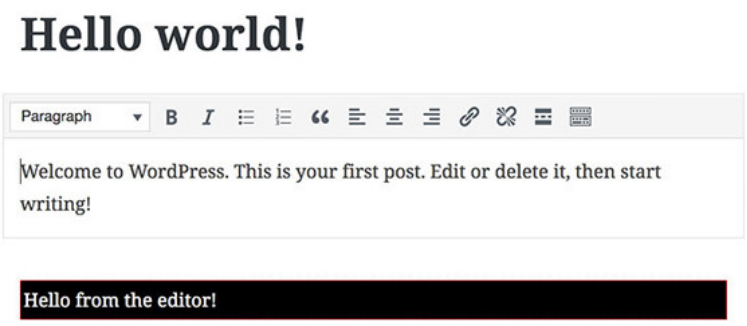
```
.
Ã ¢ Â'ÂœÃ ¢ Â'Â € Ã ¢ Â'Â € series
Ã ¢ Â'Â,Ã,Â Ã,Â Ã ¢ Â'ÂœÃ ¢ Â'Â € Ã ¢ Â'Â € block.js
Ã ¢ Â'Â,Ã,Â Ã,Â Ã ¢ Â'ÂœÃ ¢ Â'Â € Ã ¢ Â'Â € editor.css
Ã ¢ Â'Â,Ã,Â Ã,Â Ã ¢ Â'Â'Ã ¢ Â'Â € Ã ¢ Â'Â € style.css
Ã ¢ Â'Â'Ã ¢ Â'Â € Ã ¢ Â'Â € series.php
```

Download the main file of the Block in **functions.php** of the theme:

```
if (function_exists ('register_block_type')) {
require get_template_directory ().  '/blocks/series.php';
}
```

Note that we attach the download file with one condition: The block is only loaded when Gutenberg is available. This ensures compatibility with previous WordPress versions. The block will now be available in the Gutenberg interface.

This is the interface after inserting Block.



Gutenberg APIs

Gutenberg introduced two sets of APIs to register a new Block. If we look at **series.php** , we will find the following code for registering a new Block. It also loads stylesheets and JavaScripts on the front-end and the

editor.

```
register_block_type ('twentyseventeen / series', array (
'editor_script' => 'series-block-editor',
'editor_style' => 'series-block-editor',
'style' => 'series-block',
));
```

As we can see above, Block is named twentyseventeen / series, the Block name must be unique so that it does not overlap with the Block that other plugins bring.

Furthermore, Gutenberg provides a new set of JavaScript APIs to interact with the 'Block' interface in the editor. Because API is quite rich, we will focus on some specific details you should know to get a simple but effective Gutenberg block.

wp.blocks.registerBlockType

First, we will look at **wp.blocks.registerBlockType** . This function is used to register a new "Block" for the Gutenberg Editor. It requires two arguments. The first argument is that the Block name needs to follow the name registered in the **register_block_type** function on the PHP side. The second argument is an **Object that** identifies the Block properties such as title, category and some functions to render the Block interface.

```
var registerBlockType = wp.blocks.registerBlockType;

registerBlockType ('twentyseventeen / series', {
title: __ ('HTML5 Series'),
category: 'widgets',
keywords: ['html'],
edit: function (props) {},
save: function (props) {}
});
wp.element.createElement
```

This function allows you to create elements of "Block" in the post editor. The **wp.element.createElement** function is basically removed from the **React createElement ()** function, so it accepts the same set of arguments. The first argument takes the element type, for example a paragraph, a range, or a div as follows:

```
wp.element.createElement ('div');
```

We can put the function into a variable to write shorter. For example:

```
var el = wp.element.createElement;
el ('div');
```

If you prefer to use the new ES6 syntax, you can also do it this way:

```
const {createElement: el} = wp.element;
el ('div');
```

We can also add element attributes such as class name or id to the second parameter as follows:

```
var el = wp.element.createElement;
el ('div', {
```

```
'class': 'series-html5',
'id': 'series-html-post-id-001'
});
```

Div created will not make sense without content. We can add content to the argument of the third parameter:

```
var el = wp.element.createElement;
el ('p', {
'class': 'series-html5',
'id': 'series-html-post-id-001'
}, 'This article is part of our "HTML5 / CSS3 Tutorials series" - dedicated to
```

wp.components

The wp.components contain a set of names and Gutenberg components. These components are technically React custom components including Button, Popover, Spinner, Tooltip and a host of other components. We can reuse these components into a separate Block. In the following example, we will add a button element.

```
var Button = wp.components.Button;
el (Button, {
'class': 'download-button',
}, 'Download');
```

Attributes - Properties

Attributes is a way to store data in Block. This data can be like content, color, sort, URL, etc. We can get attributes from Attributes passed on the **edit ()** function, as follows:

```
edit: function (props) {
var content = props.attributes.seriesContent;

return el ('div', {
'class': 'series-html5',
'id': 'series-html-post-id-001'
}, N?i dung );
}
```

To update Attributes, we use the **setAttributes ()** function. Normally we will change the content on certain actions such as when the button is clicked, the input is filled, the option is selected, etc. In the following example, we use it to add backup content of Block in case of an unexpected incident on **seriesContent Attribute** .

```
edit: function (props) {

if (typeof props.attributes.seriesContent === 'undefined' ||! props.attributes
props.setAttribute ({
seriesContent: 'Hello World!  Here is the fallback content. '
})
}

var content = props.attributes.seriesContent;

return [
```

```
el ('div', {
'class': 'series-html5',
'id': 'series-html-post-id-001'
}, N?i dung ),
];
}
```

Save Block

The **save ()** function works similar to the **edit ()** function, except that it defines the content of Block to save to the database. Saving Block content is quite simple, as we can see below:

```
save: function (props) {

if (! props ||! props.attributes.seriesContent) {
return;
}

var content = props.attributes.seriesContent;

return [
el ('div', {
'class': 'series-html5',
'id': 'series-html-post-id-001'
}, N?i dung ),
];
}
```

What about Gutenberg?

Gutenberg will change WordPress in a better way (or maybe worse). It allows developers to adopt a new way to develop WordPress plugins and themes. Gutenberg is just a start. Soon, the 'Block' model will be extended to other areas of WordPress such as Settings API and Widget.

Learn JavaScript Deeply is the only way to understand Gutenberg and look forward to the future of WordPress. If you are familiar with the basics of JavaScript, functions, tools, strengths and weaknesses, you will quickly catch up to Gutenberg's speed.

As mentioned, Gutenberg requires a lot of APIs, enough to do almost anything for Block. You can choose to write your Block code with a simple JavaScript, JavaScript with ES6, React syntax, or even Vue.