WordPress Playground

—ToC pattern—

—ToC pattern—

How to use WordPress Playground for interactive demos

Learn how to spin a live site that demonstrates a custom plugin, a theme adapted to feature it, and a concise manual for users—no server needed

By Ronny Shani

WordPress Playground's JavaScript API is a developer-oriented tool that opens a new world of possibilities. Supporting elaborate standalone instances makes it

particularly effective for showcasing a theme or a plugin (or both) and preparing WordPress guides that go beyond textual documentation to provide a layer of interactivity.

If you're not familiar with Playground, start by reading the previous article: Introduction to Playground: running WordPress in the browser. To make the most of this tutorial, you'd need to understand the modularity of the three APIs that power Playground and how they allow you to share themes, plugins, and even complete websites—content included.

What will you build?

The project is inspired by a common use case: demonstrating how a custom plugin coupled with a theme adapted to feature it looks and functions together. This scenario requires a manual that instructs users how and where to set the data displayed in the plugin's custom fields. Using Playground to spin up a live site with an extra step-by-step guide makes the experience much more engaging.

```
None
├── playground
│   ├── zips
│   │   ├── blue-note.zip
│   │   ├── meta-block-sidebar.zip
│   ├── blueprint.json
│   ├── index.html
├── plugins
│   │   ├── meta-block-sidebar
│   │   │   ├── PLUGIN FILES...
├── themes
│   │   ├── blue-note
│   │   │   ├── THEME FILES...
├── README.md
```

You can find the code on the GitHub repository created to accompany this tutorial. The project includes the theme and plugin source files, as well as

compressed files used by Playground. Either clone or create it manually before you continue.

Create the Blueprint

Let's start with the cornerstone of every Playground project: the `blueprint.json` file. If you haven't cloned the repository, copy and paste the following code into your file:

JavaScript

```
{
  "$schema":
"https://playground.wordpress.net/blueprint-schema.json",
  "preferredVersions": {
    "php": "latest",
    "wp": "latest"
  },
  "siteOptions": {
    "blogname": "WordPress Playground Demo"
  },
  "plugins": [
    "create-block-theme",

"https://raw.githubusercontent.com/wptrainingteam/playground-d
emo-handover/main/playground/zips/meta-block-sidebar.zip"
  ],
  "steps": [
    {
      "step": "installTheme",
      "themeZipFile": {
        "resource": "url",
        "url":
"https://raw.githubusercontent.com/wptrainingteam/playground-d
emo-handover/main/playground/zips/blue-note.zip"
      }
    },
    {
      "step": "runPHP",
```

```
      "code": "<?php require_once('/wordpress/wp-load.php');
wp_insert_post(array( 'post_title' => 'Created by a
Blueprint', 'post_content' => '<!-- wp:paragraph --><p>How do
you update the meta fields?</p><!-- /wp:paragraph --><!--
wp:list --><ol><!-- wp:list-item --><li>Open the
<strong>Settings</strong> sidebar by clicking the window icon
next to the blue <strong>Update</strong> button.</li><!--
/wp:list-item --><!-- wp:list-item --><li>Click the
<strong>Meta Block Sidebar</strong> menu (below the
<strong>Summary</strong> menu).</li><!-- /wp:list-item --><!--
wp:list-item --><li>Type the <strong>Team name</strong> and
the <strong>date</strong> the person joined the company in the
respective fields.</li><!-- /wp:list-item --><!-- wp:list-item
--><li>Click the blue <strong>Update</strong> button.</li><!--
/wp:list-item --></ol><!-- /wp:list -->','post_status' =>
'publish' ));"
    }
  ],
  "features": {
    "networking": true
  },
  "login": true,
  "landingPage": "/?p=4"
}
```

Let's take a moment to explain the different steps Playground will perform for you:

1. Once launched, Playground will create a WordPress instance, using the latest PHP and WordPress versions supported.
2. Then, it will rename the site.
3. Install and activate two plugins: Create Block Theme (available on the Plugin Directory and, therefore, fetched using its `slug`), and the custom *Meta Block Sidebar* (available in the same GitHub repository, and, accordingly, fetched from the relevant URL).
4. Install and activate the adapted version of the Blue Note theme (fetching it from the same GitHub repository).

5. Create a post with a step-by-step guide.
6. Enable networking mode.
7. Log into the site as an Administrator.
8. And, finally, open the site displaying the newly created post.

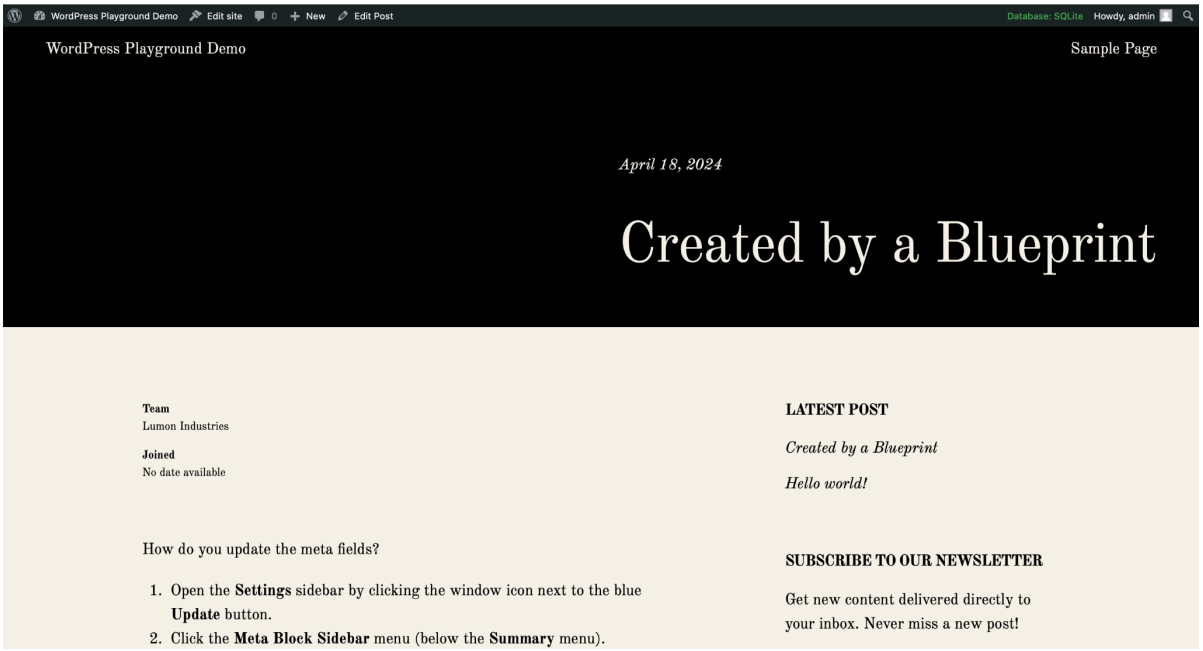
The little user manual packs content and HTML block markup together using WordPress' `wp_insert_post` function. This method works for short snippets, but it's wiser to explore Playground's more robust alternatives:

- Importing an XML file using the `importWxr` step, or
- Generating posts via WP-CLI using the eponymous `wp-cli` step.

> ℹ️ **Shorthand**
>
> The Blueprint in this tutorial takes advantage of Playground's `shorthand` syntax to make this set of instructions less verbose. The following `steps` are currently supported: `login`, `plugins`, `siteOptions`, and `defineWpConfigConsts`. Check out the documentation to learn more about how and when to use these instead of explicit `steps`.

Test the Blueprint

Now would be a good time to pause, experiment, and get comfortable with Blueprints. Copy the code above and paste it into the official Blueprint editor (still a WIP). Your instance should be identical to the original demo.



To get a concrete sense of the possibilities, try modifying some values right there in the Blueprint editor—maybe the value of `landingPage` (replace `/?p=4` with `/?p=1`) or `blogname`. If something goes wrong, undo your changes, fix any errors the editor alerts you to, and hit the **Run it** button again.

ℹ️ How to load assets in a Blueprint?

Playground needs to be able to access your files to run them. This can be done in several ways: via your custom domain, a GitHub repository, and more. The former ostensibly makes more sense, but browser security restrictions mean you'll need to handle any Cross-origin resource sharing (CORS) issues on your server. A faster way to go about it is to use GitHub's special `raw.githubusercontent.com` domain.

To prevent CORS-related errors, make your repository ***public***, and point Playground to the absolute paths of the files using the following pattern: `https://playground.wordpress.net/?blueprint-url=https://raw.githubusercontent.com/{USER}/{REPO}/{BRANCH}/PATH/TO/FILE`.

If your repository is *private*, you can only access the Playground instance via a custom domain.

Playground's in-browser editor is an invaluable tool for testing. It's also the fastest way to start adjusting the original sample to your needs. When you feel confident enough, use it to prototype your project instead of tediously tweaking the JSON on your local machine, hoping you got everything exactly right.

Feel free to alter the file names and directory structure according to your project's theme and plugins—don't forget to update the file paths and sync everything to GitHub so the files are available online (see the CORS section above). Once you're happy with the result, copy the code over to your `blueprint.json` file.

You can share the project using Playground's Query API. Here's what the URL of the example project looks like: `https://playground.wordpress.net/?blueprint-url=https://raw.githubusercontent.com/wptrainingteam/playground-demo-handover/main/playground/blueprint.json`.

Use the JavaScript API

So far, you haven't actually tapped into the JavaScript API; let's see how you can take advantage of its most powerful feature: initiating the Playground API Client. Create an `index.html` file, and copy and paste the following code:

```JavaScript
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Demo Handover Playground</title>
    <style>
      body {
        display: grid;
        place-items: center;
        background: linear-gradient(deeppink, darkorange);
```

```
      }
      header {
        font-size: 1em;
        font-family: monospace;
        display: flex;
        align-items: baseline;
        gap: 1rem;
      }
      iframe#wp-playground {
        width: 100%;
        height: 100dvh;
        border: 2px solid currentColor;
      }
    </style>
  </head>
  <body>
    <header>
      <h1>Playground Demo</h1>
      <p>Source code <a
href="https://github.com/wptrainingteam/playground-demo-handov
er" target="_blank" rel="noopener">on GitHub</a></p>
    </header>
    <iframe id="wp-playground" title="Playground Demo
Site"></iframe>
    <script type="module">
      import { startPlaygroundWeb } from
'https://playground.wordpress.net/client/index.js';
      const client = await startPlaygroundWeb({
        iframe: document.getElementById('wp-playground'),
        remoteUrl:
`https://playground.wordpress.net/remote.html`,
        blueprint:
        {
         TO-DO: ADD FINAL BLUEPRINT
        },
      });
    </script>
  </body>
</html>
```

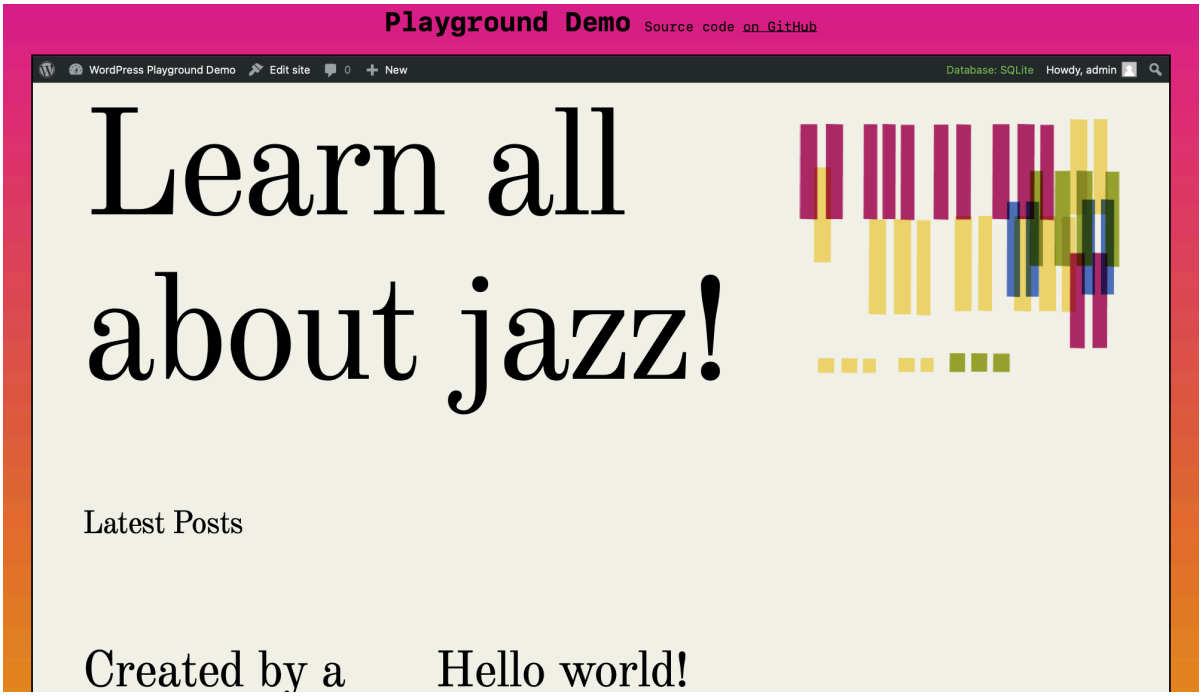
The `client` object will load inside the `iframe` via the project's `remote.html` file—the only allowed "endpoint" of this API—and run the steps provided inside the `blueprint` variable, leveraging the Blueprint API to configure the client based on your specific settings.

One of the helpful side effects of using the JavaScript API (instead of loading an `iframe`) is that you don't need to worry about CORS, absolute paths, and public repositories: the client handles everything—as long as you've set the correct URL. Paste the contents of the `blueprint.json` file created before, and feel free to change any preceding HTML markup and CSS styles to match your needs.

> ℹ️ **Live testing**
> Playground's JavaScript API uses ES Modules, which means you'll need a local server to run `index.html` in the browser. To launch a simple development server, open the terminal, and type any of the following:
> For PHP: `php -S localhost:8000`
> For NodeJS: `npx serve`

After you upload everything to GitHub and connect the repository to a custom domain and hosting provider, you can access and share the complete demo via your domain, pointing to the path of your `index.html` file (`https://www.example.com/playground/`).

This is what the original project looks like:



Your turn

Now that you're all set—with both code samples and a deeper understanding of what each line does—it's time to create your project: Upload your own modified theme or bespoke plugin, think about how to best generate posts, change the site settings, set up users (and more advanced scenarios) with `wp-cli`, and think what else you could do in the HTML-based demo itself.

How about loading multiple instances on the same page, each showing a different style variation side-by-side? Or, maybe, prepare a live tutorial teaching how to edit a template page and load a Playground in an `iframe` below, asking students to recreate it, and share their results on GitHub (using the **Export Pull Request to GitHub** option in the UI)?

The JavaScript API supports much more advanced use cases, so check out the first part of this series to see what else is possible with Playground, explore the Blueprints examples, demos, and apps for inspiration, and start playing.

Props to @bph, @zieladam, and @bjmcsherry for reviewing this post.