Guía de uso de programación: *plugin* WordPress para Mosaic

Universitat Oberta de Catalunya

Índice

Autoría: Marc Padró

PID_00267139

Primera edición: septiembre 2019
Autoría: Marc Padró

Avda. Tibidabo, 39-43, 08035 Barcelona
Realización editorial: FUOC

p5.js Embed: guía de uso

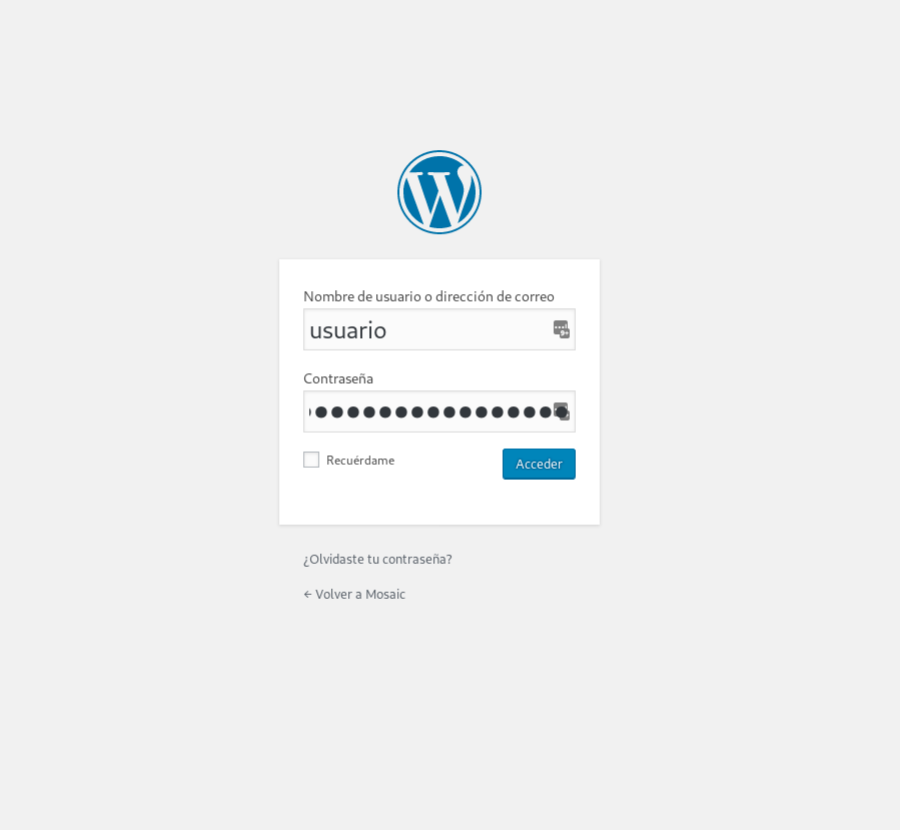
‘p5.js Embed’ es una extensión para WordPress que os permitirá incorporar código de p5.js en un sitio web que funcione con esta plataforma. Además, esta extensión permite acceder desde el código p5.js a datos sobre el sitio web y la página que se está consultando.

1. Instalar la extensión

Para instalar la extensión necesitaréis el archivo *p5js-embed.zip* disponible con los recursos de la asignatura.

Antes que nada hay que acceder al administrador del WordPress donde queramos instalarla. Para hacerlo añadimos */admin* al final de la dirección principal del sitio: por ejemplo, si la dirección de nuestro sitio es *ejemplo.com*, nos dirigiremos a ejemplo.com/admin.

Figura 1. Pantalla de inicio de sesión en WordPress



Una vez hayamos entrado nuestras credenciales (necesitaremos un usuario con permiso de administrador), accederemos al panel de control de WordPress. Debemos dirigirnos al apartado ‘Plugins’, que está en el menú situado a la izquierda de la página. Poniendo el puntero encima, accedemos al subapartado ‘Añadir nuevo’ (figura 3).

Nota

En esta guía usamos términos de WordPress de su versión en español. Si vuestra instalación está en otro idioma, los textos cambiarán, pero generalmente serán traducciones directas de los que usamos aquí. Si lo queréis, podéis cambiar el idioma de vuestra instalación en el apartado 'Opciones' del menú.

Figura 2. Pantalla inicial del panel de control de WordPress

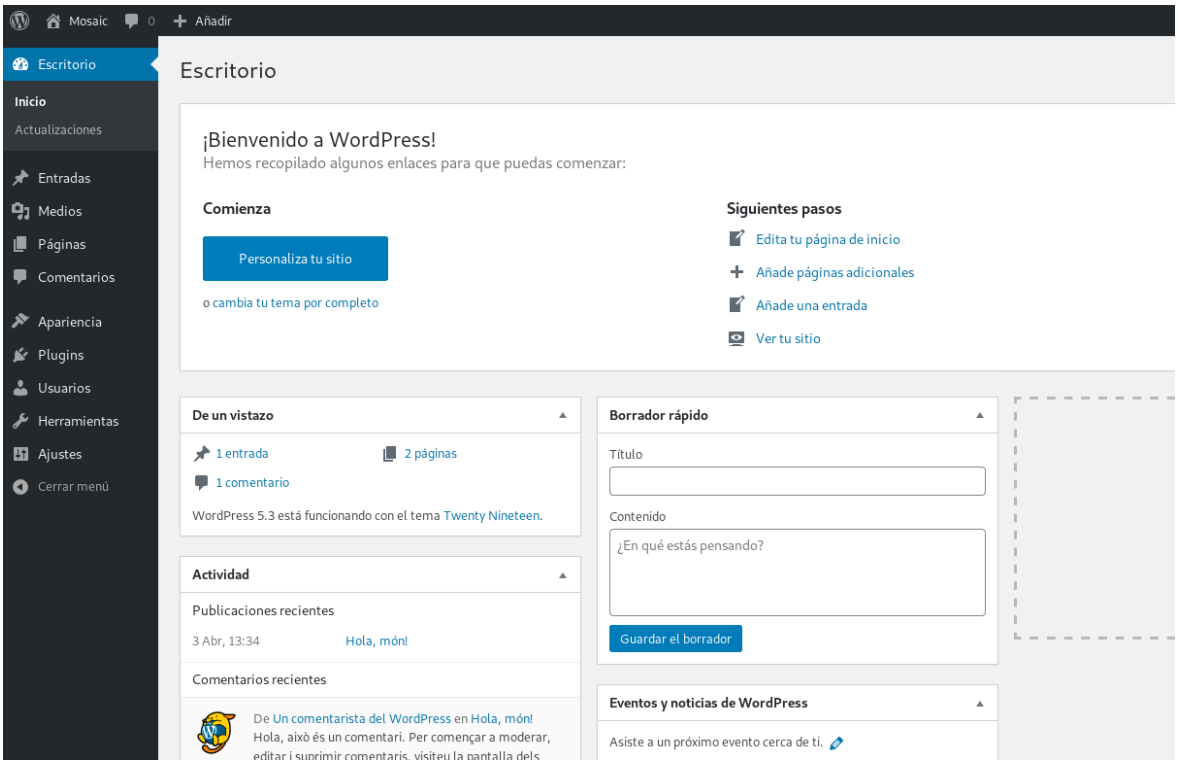
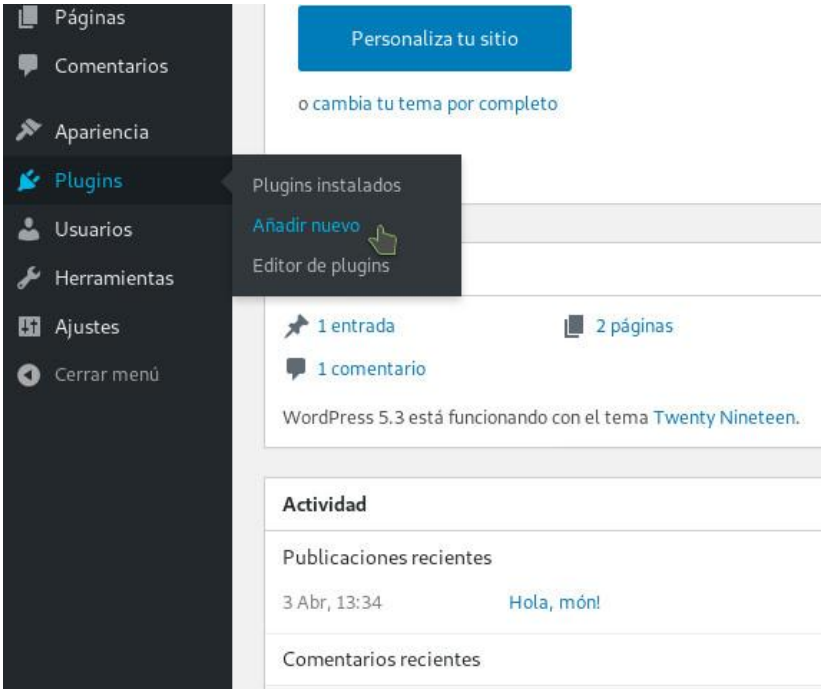


Figura 3. Apartado en el menú para añadir extensiones de WordPress



Una vez dentro, al principio de la página, junto al título, encontraremos el botón ‘Subir plugin’ (figura 4). Clicaremos en él.

Figura 4. Botón para colgar una extensión mediante un archivo de nuestro ordenador

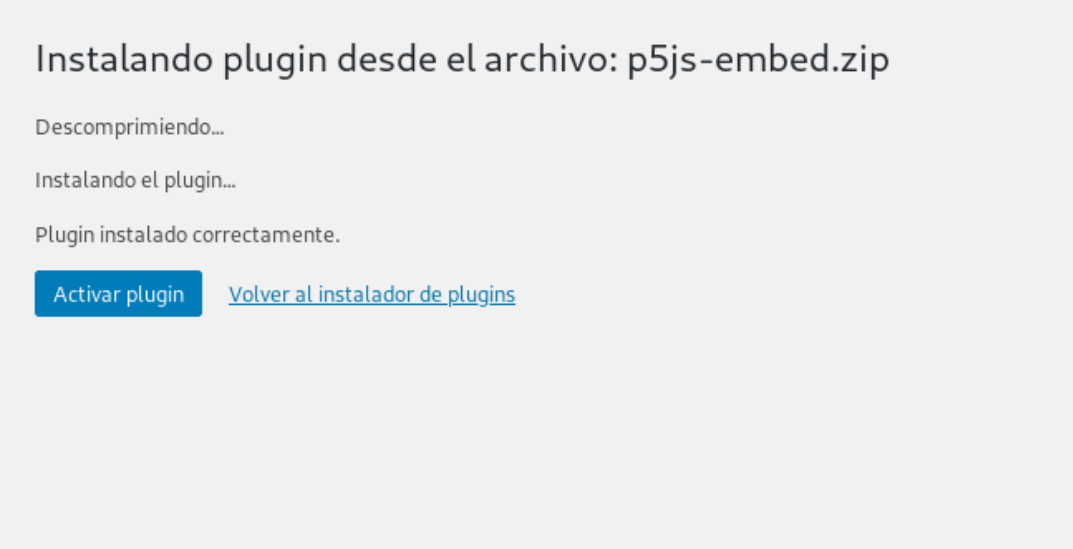
Se nos abre entonces un diálogo que nos permite subir un archivo ZIP (figura 5). Hacemos clic en el botón para buscar el archivo que tenemos que subir (el texto del botón puede variar según el navegador) y seleccionamos el archivo *p5js-embed.zip*.

Una vez seleccionado el archivo, clicamos en el botón ‘Instala ahora’.

Figura 5. Interfaz para seleccionar y subir el archivo de la extensión

Entonces nos aparecerá un texto que nos informa del progreso de la instalación. Una vez finalizada, hacemos clic en ‘Activar plugin’.

Figura 6. Texto de progreso de la instalación de la extensión



2. Incorporar el código

Una vez instalada la extensión, para incorporar el código de p5.js nos tenemos que dirigir al apartado ‘p5.js Embed’ dentro de la pestaña ‘Ajustes’ del menú (figura 7).

Figura 7. Apartado dentro del menú para acceder a la configuración de ‘p5.js Embed’

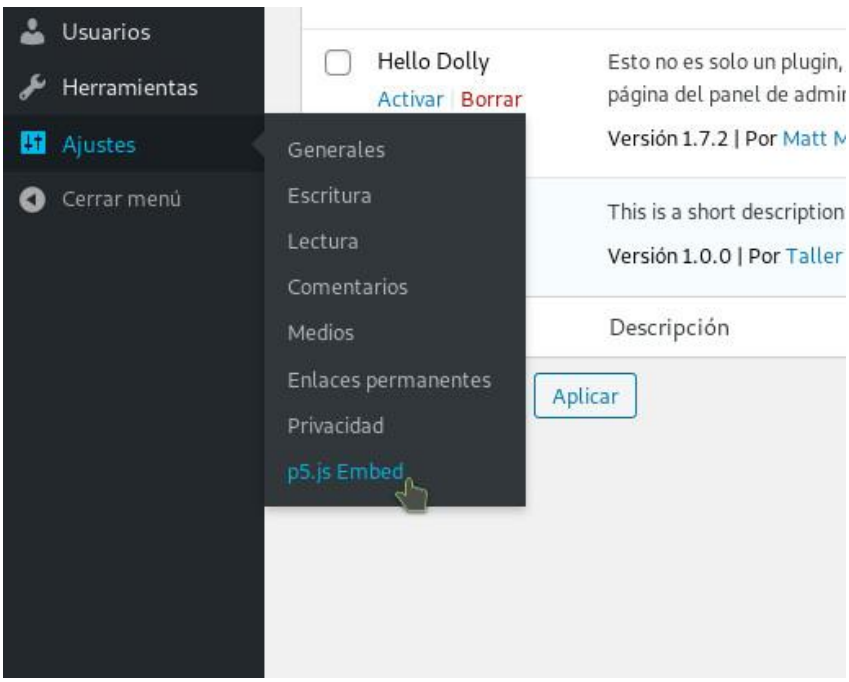
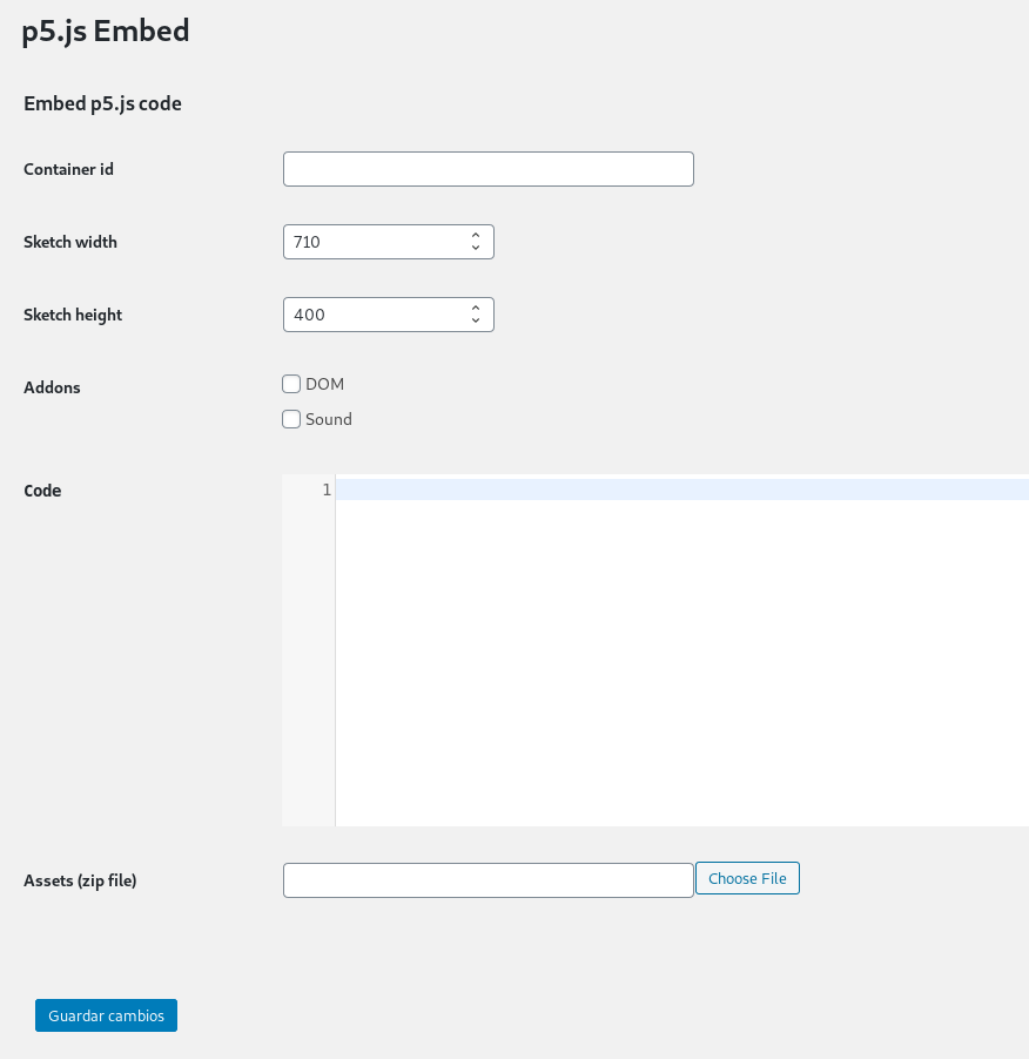


Figura 8. Opciones de configuración de ‘p5.js Embed’



Aquí encontramos una serie de opciones:

- En ‘Sketch width’ y ‘Sketch height’, tendremos que entrar la medida que tendrá nuestro *canvas*.
- Si necesitamos incorporar *addons* de p5.js, los tenemos que seleccionar en el campo ‘Addons’.
- En el campo ‘Code’ disponemos de un editor de código donde tendremos que entrar nuestro programa.
- Del campo ‘Assets’ hablaremos en el apartado 2.1.
- Del campo ‘Container id’ hablaremos en el apartado 2.2.

Tenemos que acordarnos de hacer clic en ‘Guardar cambios’ para guardar las modificaciones que hacemos.

Figura 9. Ejemplo de código en el editor de 'p5.js Embed'

```
let y = 100;

// The statements in the setup() function
// execute once when the program begins
function setup() {
  // createCanvas must be the first statement
  createCanvas(720, 400);
  stroke(255); // Set line drawing color to white
  frameRate(30);
}
// The statements in draw() are executed until the
// program is stopped. Each statement is executed in
// sequence and after the last line is read, the first
// line is executed again.
function draw() {
  background(0); // Set the background to black
  y = y - 1;
  if (y < 0) {
    y = height;
  }
  line(0, y, width, y);
}
```

Mientras no necesitemos integrarlo en el WordPress, nos será más cómodo trabajar el código en un archivo aparte, como con cualquier otro proyecto de p5.js. Cuando queramos ver el resultado en contexto, copiamos y enganchamos el programa en el editor de WordPress. Si necesitamos hacer retoques o queremos hacer algunas pruebas con la información de WordPress que la extensión nos proporciona (encontraréis más información sobre este tema en el apartado 3), entonces quizás preferiremos editar el código directamente en este editor.

Solo disponemos de un campo de texto para nuestro código, así que lo tendremos que agrupar todo en un único espacio. Es recomendable, pues, separar diferentes secciones con comentarios que nos permitan distinguirlas fácilmente.

2.1. Incorporar otros recursos

Puede ser que en nuestro programa queramos incorporar imágenes, tipografías, documentos con datos o cualquier otro tipo de documento. Para hacerlo, disponemos de un campo 'Assets', donde podemos seleccionar un archivo ZIP que contenga todos los recursos que queremos incorporar.

Si hacemos clic en el botón 'Choose file, se nos abre una ventana. En la pestaña 'Subir ficheros' (figura 10) podemos seleccionar un nuevo archivo para colgar, y en la pestaña 'Biblioteca de medios' (figura 11), un archivo que hayamos subido anteriormente. Esto nos permite subir archivos con diferentes versiones de los recursos que queremos usar y alternarlos. Si queremos modificar alguno de los recursos que hemos subido o añadir más, sin embargo, tendremos que subir de nuevo un fichero con todos los recursos.

Figura 10. Pestaña para colgar un nuevo archivo con recursos

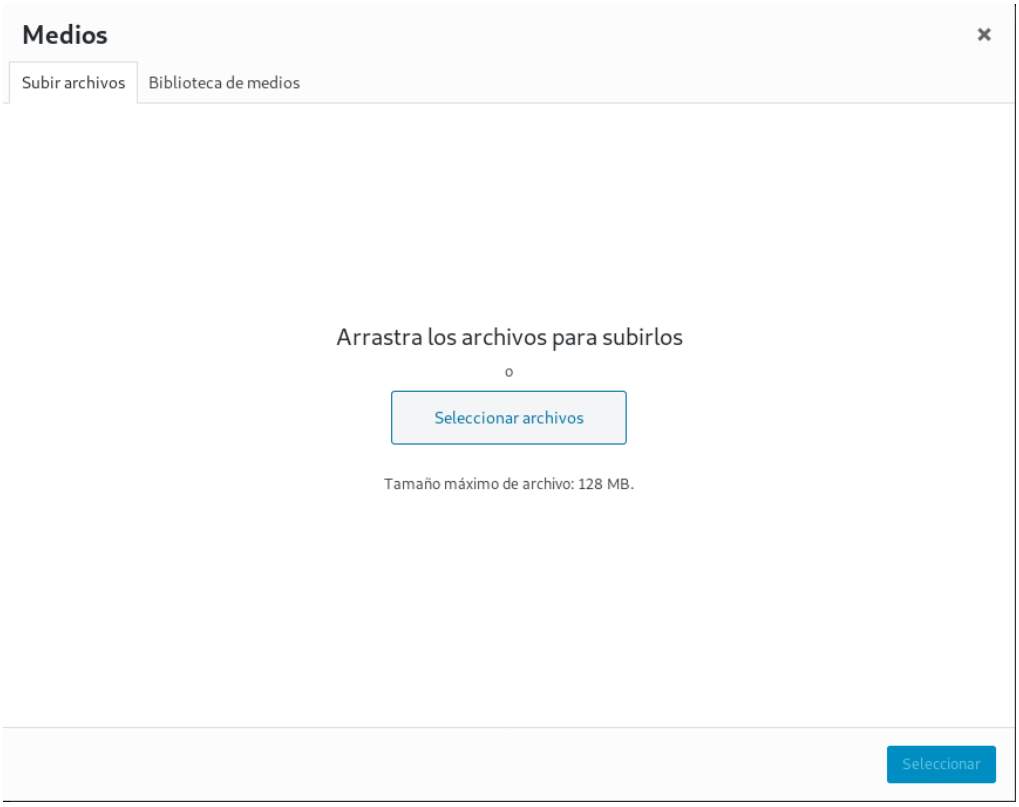
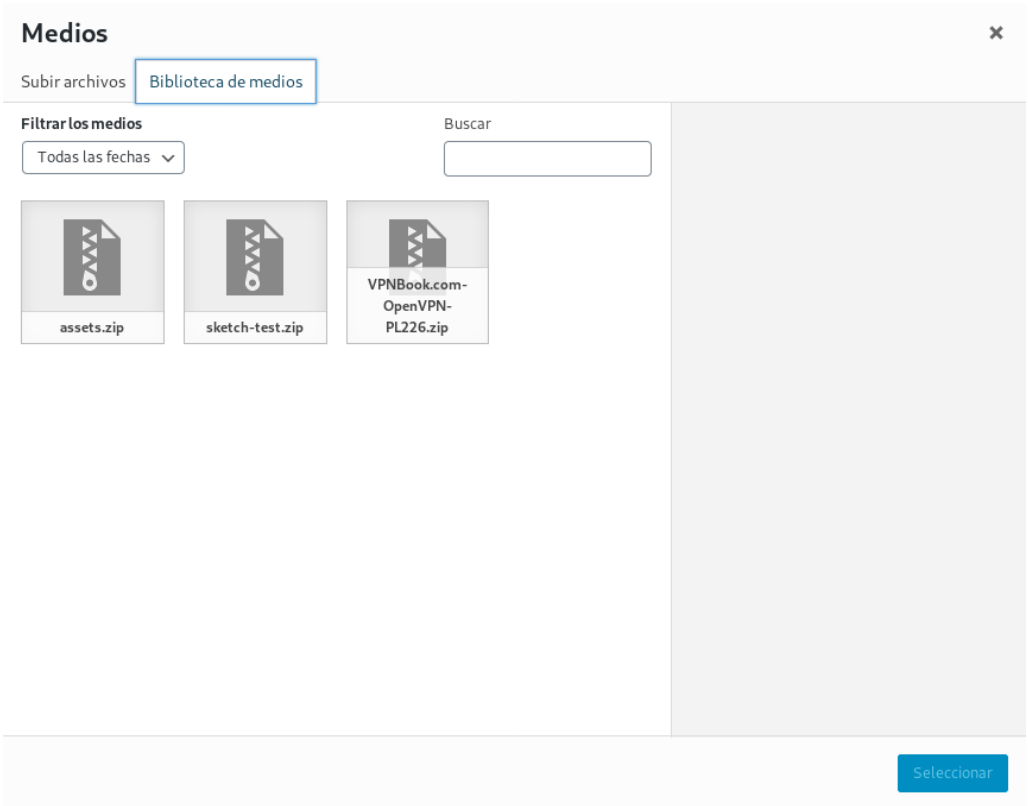


Figura 11. Pestaña para seleccionar un archivo de recursos que hayamos subido previamente



A la hora de comprimir en un fichero ZIP los archivos que queremos subir, tenemos que tener en cuenta que hay que respetar la estructura de carpetas que usamos dentro del código para acceder a los documentos. Es decir, si tenemos una imagen *fondos.png* dentro de un directorio *recursos* a la que accedemos desde nuestro código con la ruta "*recursos/fondos.png*" (ej: *var fondos = loadImage("recursos/fondos.png");*) tendremos que comprimir la carpeta *recursos* y no solo los ficheros que contiene. De este modo se respetará la misma ruta de acceso cuando subimos el fichero.

Igualmente, si los recursos tienen que estar en el directorio principal de nuestro proyecto (ex: *var fondos = loadImage("fondo.png");*), comprimiremos directamente todos los documentos que queramos subir en un mismo archivo ZIP.

Por comodidad, recomendamos que dispongáis de todos los recursos que tengáis que usar en vuestro programa en un directorio específico y que lo comprimáis cuando tengáis que subir los documentos.

2.2. Mostrar nuestro programa

Una vez hayamos introducido el código de nuestro programa, tendremos que mostrarlo en algún lugar. Tenemos dos opciones para hacerlo:

1. Usar el atributo *id* de un elemento HTML dentro del cual lo queremos incrustar.

2. Añadir un *shortcode* o 'Código de sustitución' en cualquier página o *post* de nuestro WordPress.

2.2.1. Usar el *id* de un elemento HTML

A pesar de que esta opción es más adecuada si queremos integrar el programa en el diseño general del sitio web, hay que estar un poco familiarizado con el funcionamiento HTML y con los temas de WordPress. Si no es el caso, es más sencillo optar por el *shortcode*.

Si nuestro tema de WordPress dispone de un elemento HTML dentro del cual queremos incrustar nuestro *sketch*, podemos hacerlo si este elemento dispone de un atributo *id* o si nosotros mismos somos capaces de añadirlo. Entonces solo debemos introducir este *id* en el campo 'Container id' de las opciones de 'p5.js Embed'. De este modo, nuestro *sketch* se incrustará al final del contenido de este elemento. Si el tema tiene la estructura adecuada, esto nos permite incorporar nuestro programa dentro de alguno de los elementos comunes del sitio web, como puede ser la cabecera. Generalmente necesitaremos un tema que ya esté adaptado para esta finalidad, o ser capaces nosotros de adaptarlo, puesto que, si no, seguramente no dispondremos del espacio o de la distribución necesarios para incorporar nuestro *sketch.*

Figura 12. Si nuestra página dispone de un elemento HTML con un atributo *id*, podemos incrustar nuestro *sketch*. En este caso el *id* 'logo'

Figura 13. Ejemplo de elemento HTML con el *id* 'logo' donde podríamos incrustar nuestro *sketch*

```
<div class="site-branding">
    <p class="site-title"><a href="http://work.lcl/test/" rel="h
    <p class="site-description">
        Otro sitio gestionado con WordPress
    </p>

    <div id="logo">
        <!-- Este elemento contendrá nuestro Sketch -->
    </div>

    <nav id="site-navigation" class="main-navigation" aria-label
        <div class="menu-menu-container">
            <ul id="menu-menu" class="main-menu">
                <li id="menu-item-16" class="menu-item menu-item
                    <a href="http://work.lcl/test/" aria-current
                </li>
```

2.2.2. Usar un *shortcode*

Un *shortcode* es un pequeño código que nos permite incrustar determinados elementos dentro del contenido de las páginas y de los *posts* de WordPress.

Nota

Los *posts* de WordPress se denominan 'Entradas' en español, pero en este texto usaremos el término en inglés.

Los *shortcodes* se escriben entre corchetes ([ ]). Si queremos incrustar nuestro *sketch* en el contenido de cualquier página o *post*, solo hace falta que incluyamos el código *[p5js_embed]*. Podemos escribirlo directamente en el texto de la página (figura 14) o, si usamos el editor de bloques de las últimas versiones de WordPress, podemos añadir un bloque 'Shortcode', donde escribiremos el mismo código (figura 15).

Figura 14. *Shortcode* escrito directamente en el texto de una página

> Alcor, tengo un perro que se llama Firulais y me gusta el rebujito. (Y las tardes largas con café).

…o algo así:

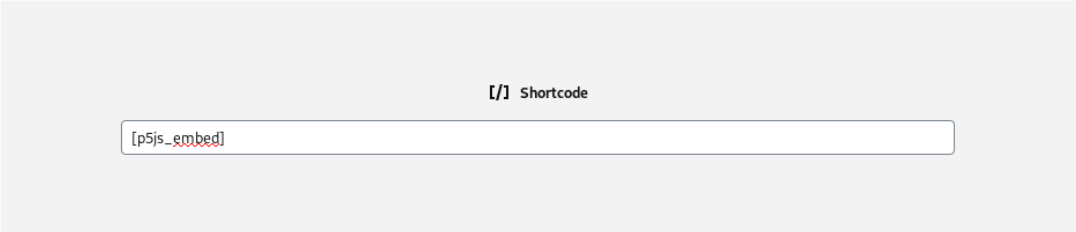
[p5js_embed]

> La empresa «Mariscos Recio» fue fundada por Antonio Recio Mata. Empezó siendo una pequeña

Figura 15. *Shortcode* escrito dentro de un bloque 'Código de sustitución'

> Alcor, tengo un perro que se llama Firulais y me gusta el rebujito. (Y las tardes largas con café).

…o algo así:

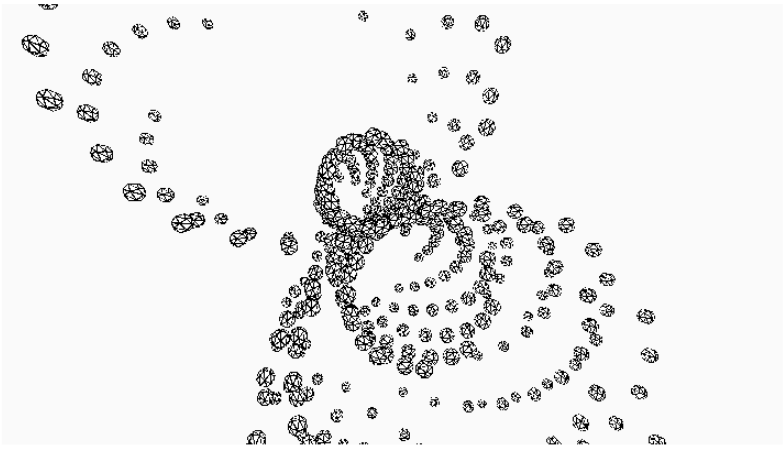


> La empresa «Mariscos Recio» fue fundada por Antonio Recio Mata. Empezó siendo una pequeña

Figura 16. Resultado: *Sketch* incrustado en la página, allá donde hemos incluido el *shortcode*

> Alcor, tengo un perro que se llama Firulais y me gusta el rebujito. (Y las tardes largas con café).

…o algo así:



> La empresa «Mariscos Recio» fue fundada por Antonio Recio Mata. Empezó siendo una pequeña em-

3. Acceder a los datos sobre el sitio web

'p5.js Embed' nos facilita una serie de datos sobre el sitio web WordPress a los cuales podemos acceder desde nuestro código de p5.js. Para hacerlo, solo debemos consultar la variable *wp_data*, que es definida por la extensión antes de que se ejecute nuestro programa.

wp_data es un objeto que consta de las propiedades siguientes:

- **site_title:** (*String*) Título del sitio web.
- **site_description:** (*String*) Descripción del sitio web.
- **post_count:** (*Number*) Número total de posts publicados en el sitio.
- **first_post:** (*Object*) Primer *post* que se publicó. Es un objeto con la estructura siguiente:
  - **id:** (*Number*) Identificador numérico único para cada *post*.
  - **title:** (*String*) Título del *post*.
  - **date:** (*Number*) Fecha de creación del *post* en milisegundos desde la época Unix (1 de enero de 1970 a las 00.00.00 h). Pensado para usar con la clase Date de Javascript.
  - **type:** (*String*) Tipo de *post*. En este caso será siempre «post».
  - **author:** (*Object*) Objeto con la información siguiente sobre el autor del *post*:
    - **id:** (*Number*) Identificador numérico único para cada autor.
    - **name:** (*String*) Nombre del autor.
    - **count:** (*Number*) Número *de posts* publicados por este autor.
  - **categorías:** (*Array*) *Array* de las categorías asignadas al *post*. Cada elemento del *array* tiene la estructura siguiente:
    - **id:** (*Number*) Identificador numérico único para cada categoría.
    - **name:** (*String*) Nombre de la categoría.
    - **count:** (*Number*) Número *de posts* a los cuales se ha asignado esta categoría.

        - **type:** (*String*) Tipo de taxonomía. En este caso será siempre «category».
        - **parent:** (*Number*) Identificador numérico de la 'categoría madre' de esta. Las categorías son jerárquicas y, por lo tanto, puede ser que una categoría sea subcategoría de otra. En caso de que se trate de una categoría de primer nivel, este valor será 0.
    - **tags:** (*Array*) *Array* de los *tags* ('etiquetas' en español) asignados al *post*. Tiene la misma estructura que 'categories', pero en este caso 'type' es siempre «post_tag» y 'parent' es siempre 0.

- **last_post:** (*Object*) Último *post* que se ha publicado. Tiene la misma estructura que 'first_post'.
- **categorías:** (*Array*) Lista de todas las categorías que hay. Tiene la misma estructura que la propiedad 'categories' de 'first_post' y 'last_post'.
- **tags:** (*Array*) Lista de todos los *tags* que hay. Tiene la misma estructura que la propiedad 'tags' de 'first_post' y 'last_post'.
- **current:** (*Object*) Contiene información sobre la página actual. Sus propiedades pueden variar según el tipo de contenido que estemos visualizando. Estas son las posibilidades:
    - **type:** (*String*) Nos indica de qué tipo de contenido se trata. Las opciones son:
        - **«posts_page»:** Es la página con **la lista *de posts*** publicados.
        - **«post»:** Es la página específica de **un único *post***.
        - **«page»:** Es una **página genérica**.
        - **«category»:** Es la lista *de posts* para una **categoría** concreta.
        - **«post_tag»:** Es la lista *de posts* para un **tag** concreto.
        - **«author»:** Es la lista *de posts* de un **autor**.
        - **«date»:** Es la lista *de posts* de una **fecha o rango de fechas** concretas.
        - **«404»:** Significa que la dirección a la cual se ha accedido no se ha encontrado, porque no se corresponde con ninguna página del sitio.
    - **page:** (*Number*) Nos indica en qué número de página del contenido que estamos visualizando nos encontramos, puesto que puede ser que las listas *de posts* o los contenidos de las páginas se extiendan a lo largo de varios números de página. Empieza a contar desde 1, pero si marca 0, quiere decir que no se ha especificado ningún número de página, y por lo tanto, equivale a 1.

- **page_title:** (*String*) Título de la página tal como aparece en la pestaña o en la ventana del navegador. Solo la parte que hace referencia al título de la página actual.

- **tab_title:** (*String*) Título completo que aparece en la pestaña o en la ventana del navegador. Generalmente tiene un formato tipo «Título de la página - Título del sitio», mientras que en la página principal del sitio acostumbra a ser tipo «Título del sitio - Descripción del sitio».

- **is_home:** (*Boolean*) Nos indica si la página actual es la página principal del sitio (la portada) o no.

- **posts:** (*Array*) Solo aparece si la página actual es una lista de *posts*. Contiene un 'Array' con todos los *posts* de la lista. Cada uno de los elementos tiene la misma estructura que 'first_post' o 'last_post'.

Aparte de las propiedades ya descritas, cuando 'current' sea de tipo **«post»** o «**page»** tendrá las mismas propiedades que 'first_post' o 'last_post', referentes al *post* o la página actual.

Además, si es de tipo **«page»** tendrá también una propiedad **'parent**' con el identificador numérico de la página superior (o 0 si no hay), puesto que las páginas pueden ser jerárquicas.

Si 'current' es de tipo **«category»** o «**post_ tag»**, tendrá las mismas propiedades que los elementos de los 'Arrays' 'categories' o 'tags'.

Si es del tipo **«author»**, tendrá las mismas propiedades que la propiedad 'author' de 'first_post' o 'last_post'.

Si 'current' es de tipo **«date»**, tendrá una propiedad '**vars**', que es un objeto cuyas propiedades pueden ser una o más de las siguientes: '**year**', '**month**', '**day**', '**hour**', '**minute**' y '**second**'. La fecha de publicación de los *posts* de la lista coincidirá con los valores de estas variables temporales (las que estén presentes). Por ejemplo, si estamos consultando una lista *de posts* publicados en abril del 2015, 'vars' tendrá el formato siguiente: { year: 2015, month: 4 }.

Así pues, si queremos obtener el número total de *posts* publicados, consultaremos ***wp_data.post_count***. Si sabemos que estamos en una página que es una lista de *posts* (porque, por ejemplo, ***"date" === wp_data.current.type***) y queremos saber cuántos se muestran en la página actual, podemos consultar ***wp_data.current.posts.length***. O si queremos saber cuántos *posts* ha publicado el autor del último *post*, lo encontraremos en ***wp_data.last_post.author.count***.

Os recomendamos que, si no os queda clara la estructura de estos datos, o si queréis consultar toda la información para una página concreta, hagáis ***console.log(wp_vars)*** al principio de vuestro código para mostrar el contenido en la consola del navegador.