How to Embed Random Content Inside a Grid Layout (With PHP)

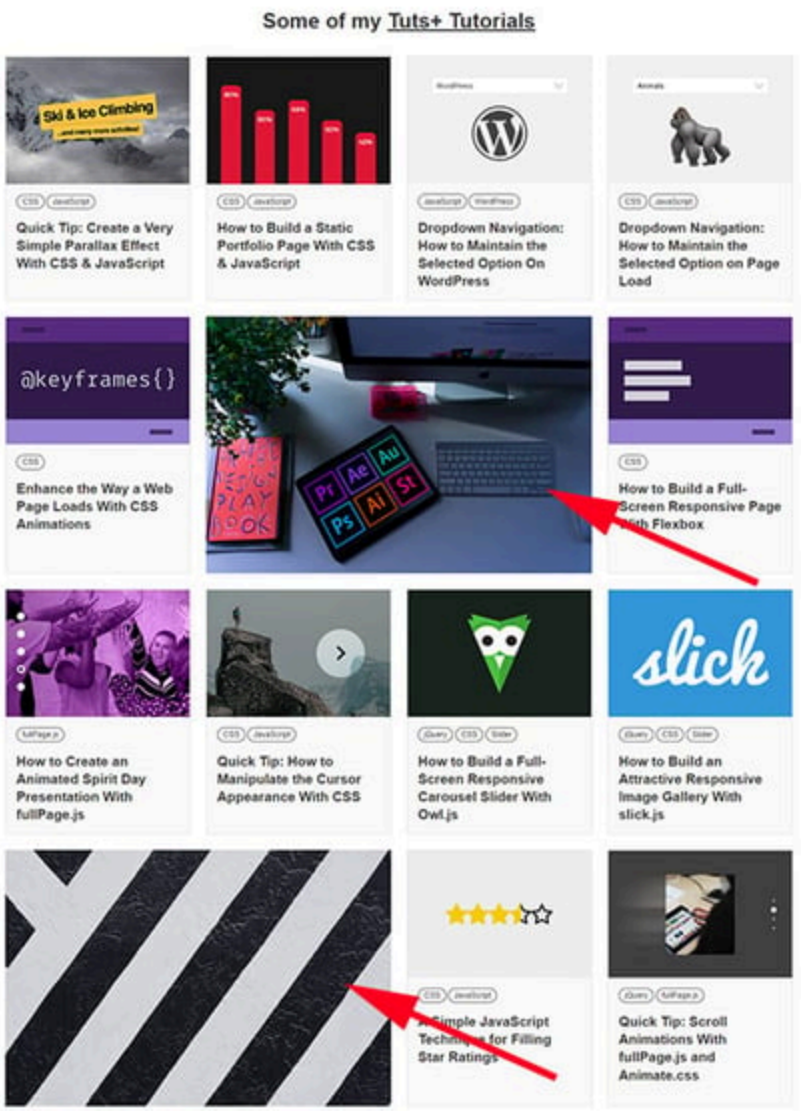
George Martsoukos

Jan 9, 2023 • 6 min read

HTML/CSS HTML WordPress

In this tutorial, we'll learn how to embed random content inside a grid layout, a popular technique used in different sites to catch visitors' eyes and promote their products/services.

This is the perfect way to display ads, or promo blocks for your own content. It also gives your layouts an interesting visual break from repetitive grids.



As usual, to better understand what we're going to build, there's an accompanying demo. But, as this demo uses some PHP code, it needs a server to run.

You can download the project files from this GitHub repository.

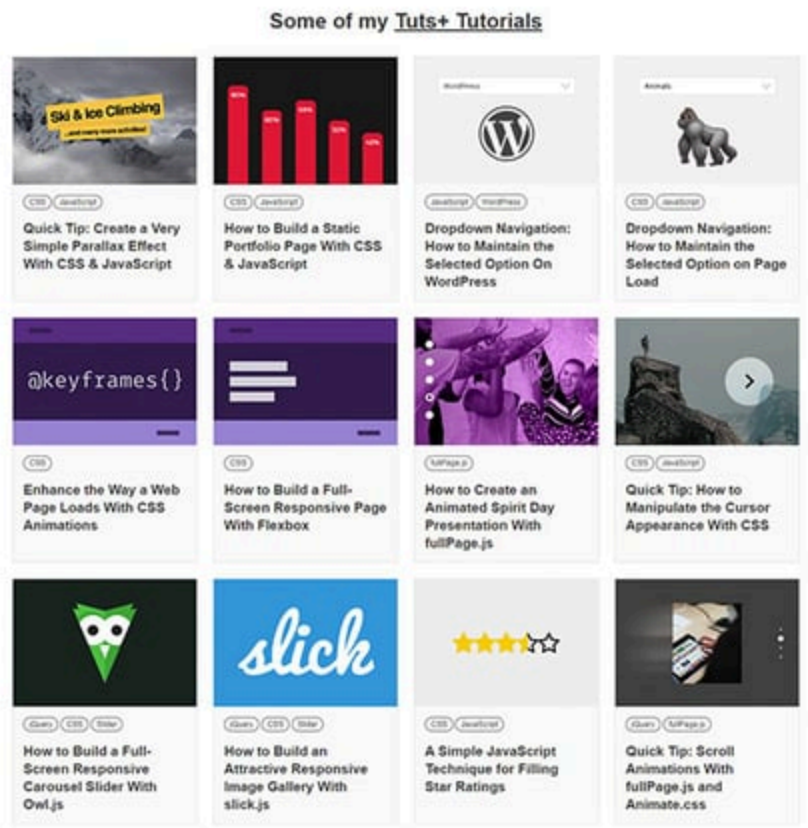
As soon as you're able to run the demo through a server, notice two things:

The embedded banners
and as you reload the page, the banner images change.

> This tutorial assumes that you're familiar, to some degree, with a server-side language like PHP.

Understanding the Layout

Each instructor here at Tuts+ has their own archive page. In a previous tutorial, we recreated a tutorial list like this with our own markup.



The Markup

We used straightforward HTML to build the structure. This time, let's automate things and store the posts inside a PHP array. Here we'll go with PHP, but regardless of the language or the CMS/framework, the logic will remain the same. We should loop through the source and retrieve the posts.

Our posts' format will look as follows:

```
$articles      = array(
  array(
    'title'      => 'Quick Tip: Create a Very Simple Parallax Effect
    'image'      => 'https://s3-us-west-2.amazonaws.com/s.cdpn.io/16
    'link'       => 'https://webdesign.tutsplus.com/tutorials/quick-
    'categories' => array(
      'CSS',
      'JavaScript',
    ),
  ),
  array(
    'title'      => 'How to Build a Static Portfolio Page With CSS &
    'image'      => 'https://s3-us-west-2.amazonaws.com/s.cdpn.io/16
    'link'       => 'https://webdesign.tutsplus.com/tutorials/how-to
    'categories' => array(
```

```
        'CSS',
        'JavaScript',
      ),
    ),

    // more articles here
);
```

And here's our loop logic:

```
<?php if ( ! empty( $articles ) ) : ?>
  <div class="container">
    <ol class="posts">
      <?php foreach ( $articles as $key => $article ) : ?>
        <li class="post">
          <article>
            <figure>
              <a href="<?php echo $article['link']; ?>" target="_bla
                <img width="300" height="208" src="<?php echo $artic
              </a>
              <figcaption>
                <ol class="post-categories">
                  <?php foreach ( $article['categories'] as $cat ) :
                    <li>
                      <a href="">
                        <?php echo $cat; ?>
                      </a>
                    </li>
                  <?php endforeach; ?>
                </ol>
                <h2 class="post-title">
                  <a href="<?php echo $article['link']; ?>" target="
                    <?php echo $article['title']; ?>
                  </a>
                </h2>
              </figcaption>
            </figure>
          </article>
        </li>
      <?php endforeach; ?>
    </ol>
  </div>
<?php endif; ?>
```

As said, depending on the language or CMS you're going to use, things will change. For example, WordPress has a built-in loop for all primary queries.

```
<!-- Start the Loop. -->
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>

  <!-- Add post info using WP built-in functions -->

```

```php
<?php endwhile; else : ?>

  <!-- The very first "if" tested to see if there were any Posts to
  <!-- display. This "else" part tells what do if there weren't any.
  <p><?php esc_html_e( 'Sorry, no posts matched your criteria.' ); ?>

  <!-- REALLY stop The Loop. -->
<?php endif; ?>
```

> In a real-world example, it's always wise to secure the data output. Each language/framework/CMS has its own functions for this purpose. For example, this page showcases the built-in WordPress functions.

The Styles

Apart from the markup, we'll also keep most of the styles from the previous tutorial. We'll only make a few changes to make the layout responsive.

Here are all the styles:

```css
:root {
  --black: #3a3a3a;
  --white: #fff;
  --green: #49b293;
}

* {
  margin: 0;
  padding: 0;
}

img {
  display: block;
  max-width: 100%;
  height: auto;
}

ol {
  list-style: none;
}

a {
  text-decoration: none;
  color: inherit;
}

body {
```

```
    margin: 50px 0;
    color: var(--black);
    font: 1rem/1.3 sans-serif;
  }

  .container {
    max-width: 1200px;
    padding: 0 15px;
    margin: 0 auto;
  }

  h1 {
    text-align: center;
    margin-bottom: 2rem;
  }

  h1 a {
    text-decoration: underline;
  }

  .posts {
    display: grid;
    grid-gap: 1.5rem;
  }

  .posts .post {
    width: 300px;
    margin: 0 auto;
    border: 1px solid rgba(0, 0, 0, 0.1);
  }

  .posts > li {
    background: #fafafa;
  }

  .posts .post-title {
    font-size: 1.3rem;
  }

  .posts .post-title:hover {
    text-decoration: underline;
  }

  .posts figcaption {
    padding: 1rem;
  }

  .posts .post-categories {
    margin-bottom: 0.75rem;
    font-size: 0.75rem;
  }

  .posts .post-categories * {
    display: inline-block;
  }

  .posts .post-categories li {
    margin-bottom: 0.2rem;
  }

  .posts .post-categories a {
```

```
    padding: 0.2rem 0.5rem;
    border-radius: 1rem;
    border: 1px solid;
    line-height: normal;
    transition: all 0.1s;
  }

  .posts .post-categories a:hover {
    background: var(--green);
    color: var(--white);
  }

  @media (min-width: 500px) {
    .posts {
      grid-template-columns: repeat(2, 1fr);
    }

    .posts .post {
      width: auto;
    }
  }

  @media (min-width: 600px) {
    .posts {
      grid-template-columns: repeat(3, 1fr);
    }
  }

  @media (min-width: 900px) {
    .posts {
      grid-template-columns: repeat(4, 1fr);
    }
  }
```

Embed Banners

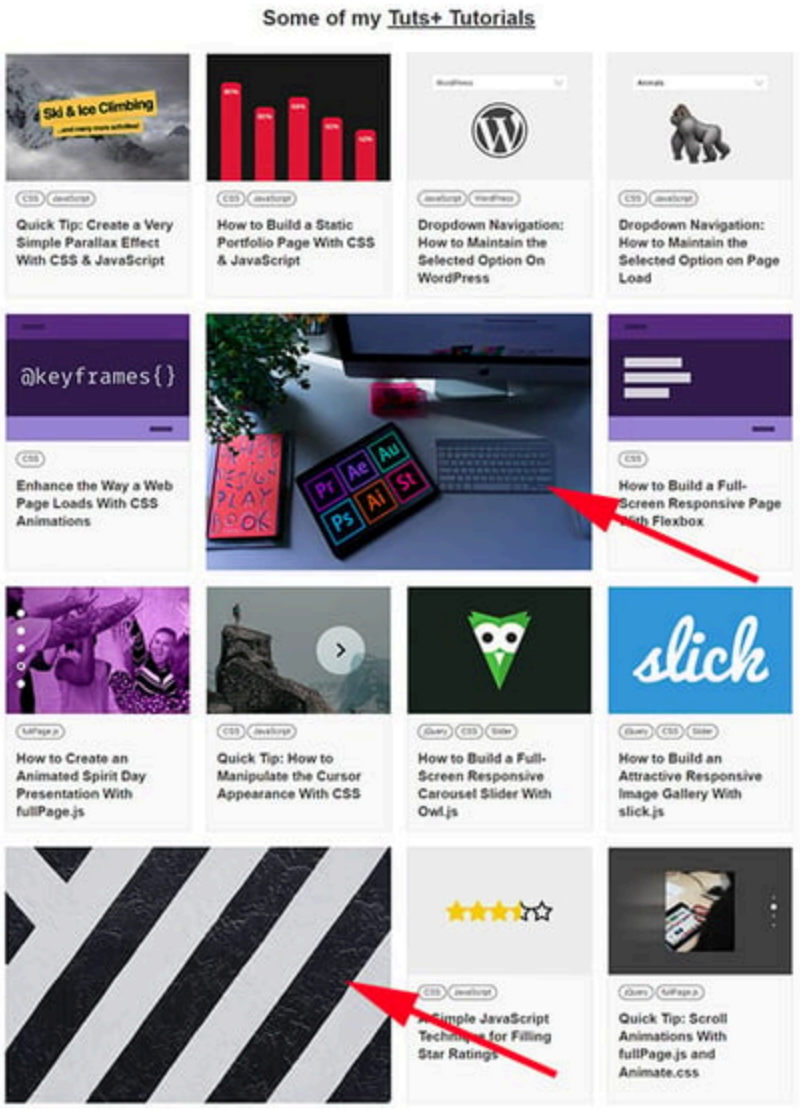
Let’s now assume that we want to place banners inside our grid. In this case, we’ll just use image banners, but in your case, you can insert ads coming from different sources, carousels, videos, or anything else you like.

Our banners have to satisfy the following requirements:

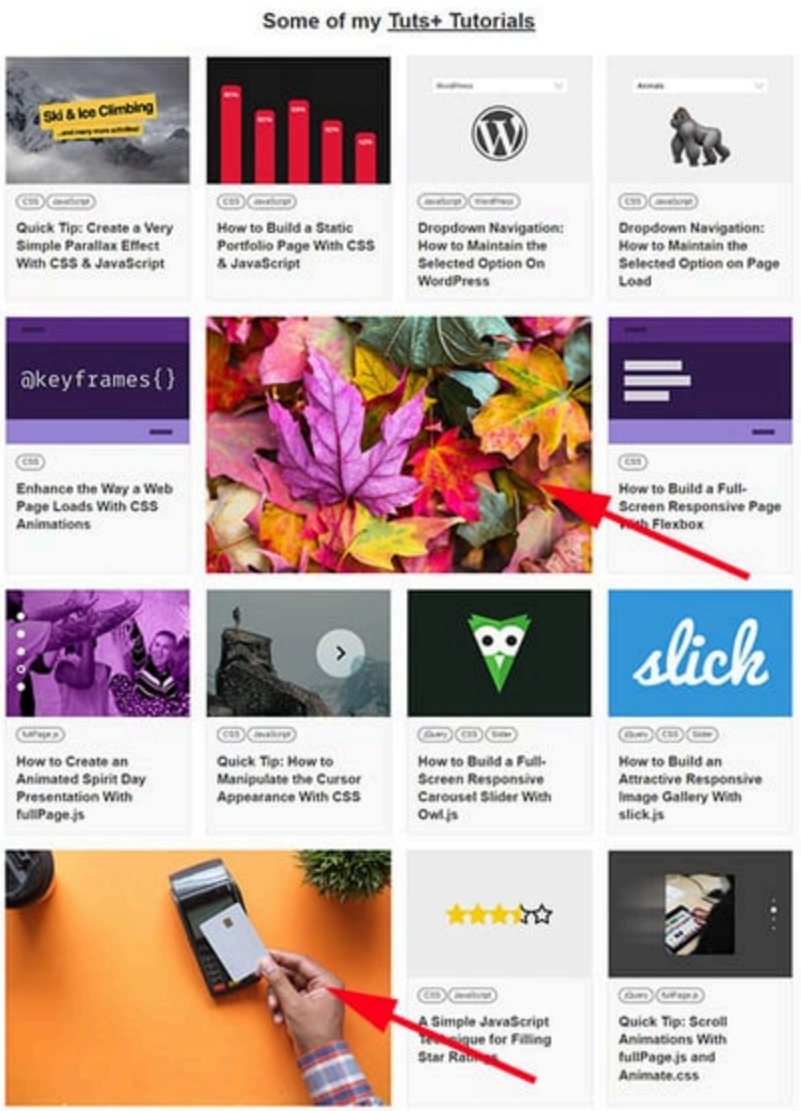
They should come after every fifth column. In our case, there are 12 posts, so our grid will contain two banners. Of course, in your case, you can have many more.
On each page load, they should appear randomly, meaning that there won’t be any fixed places for certain banners.

Plus, the embedded banners should be unique, meaning that a single banner won't ever appear twice inside the same page.

Here's an example of the layout we want to produce:



As mentioned, banners will appear randomly, so on another page load, we might see different ones, like this:



To accomplish this behavior, we'll use different PHP array functions (`array_keys`, `array_diff`, `array_rand`, `array_push`).

Let's take note of the steps we'll follow:

Store inside the `$images` array all the banners (coming from Unsplash) and grab the keys from this array.
Initialize the `$exclude_images_keys` array, where we'll add the key of each banner that is being added to the grid. By default, it'll be empty.
Inside the loop, we'll check to see if the current element's index isn't 0 and is divisible by 5.
If that happens, we'll compare the `$images_keys` and `$exclude_images_keys` arrays. We'll return their unique values, if any, and store them inside the `$in_items` array.

Grab a random key from the `$in_items` array.
Add this key to the `$exclude_images_keys` array to exclude the associated banner from future selections.
Pick the banner with this key and place it in the grid.

Here's the PHP code responsible for this functionality:

```php
<?php
// 1
$images              = array(
  'banner1.jpg',
  'banner2.jpg',
  'banner3.jpg',
  'banner4.jpg',
  'banner5.jpg',
  'banner6.jpg',
);
$images_keys         = array_keys( $images );
$exclude_images_keys = array();

foreach ( $articles as $key => $article ) :
  // 3
  if ( 0 !== $key && 0 === $key % 5 ) :
    // 4
    $in_items         = array_diff( $images_keys, $exclude_images_key
    // 5
    $random_image_key = array_rand( $in_items );
    // 6
    array_push( $exclude_images_keys, $random_image_key );
    // 7
    $random_image = $images[ $random_image_key ];
    ?>
    <li class="banner">
      <img width="800" height="533" src="img/<?php echo $random_image
    </li>
    <?php
  endif;
  ...
 endforeach;
```

And the additional CSS for our banners:

```css
.posts .banner img {
  width: 100%;
  height: 100%;
  object-fit: cover;
}

@media (min-width: 900px) {
  .posts .banner {
    grid-column: span 2;
  }
```

```
11  }
```

Conclusion

That's all, folks! I hope you enjoyed this little exercise as much as I did and that it gave you some ideas on how to embed advertising items just within a grid layout. Also, by using the modulo operator (`%`) in the way we covered here, you can create multi-column layouts with dynamic patterns.

As always, thanks a lot for reading!

By **George Martsoukos**

Like