



# 8 Buenas Prácticas Para Perfecta Documentación CSS

by [Adriana De La Cuadra](#) 2 Jan 2018

Length: Long Languages: Español ▼

Documentation

CSS



Spanish (Español) translation by [Jean Perez](#) (you can also [view the original English article](#))

En el mundo del CSS, documentación es subutilizada. Puesto que la documentación no es visible para el usuario final, su valor es a menudo pasado por alto por los clientes. También, si es tu primera vez documentación de código, puede ser difícil determinar *qué* documento y *cómo* hacerlo más eficaz.

Sin embargo documentar CSS puede ofrecer mucho a su proyecto: de fomentar mejores prácticas de código para facilitar la inducción de nuevos miembros del equipo. En este artículo explica las ventajas de CSS de documentar y compartir lo que mi equipo y yo en [Bitovi](#) consideran las mejores prácticas para trabajar documentación para usted, no al revés. Entremos en él.

## 1. Establecer las Reglas

Es difícil subirse al carro de documentación cuando no está claro para usted y su equipo qué documento o cómo desea que funcione. Así que el primer paso es ponerse de acuerdo sobre los **convenios** que **vamos** a **usar** y cómo **debe implementarlos**. Usted puede hacer esto en un documento vivo para que todos en el equipo pueden contribuir. De esta manera, su enfoque cambia o se convierte en la más completa, usted puede mantenerla actualizada. Google doc, una página wiki en su repositorio de código o (mejor aún) una página compartida en su "vida guía de estilo" son todos buenos lugares para ello.

Ahora vamos a mira la "reglas del juego" que puede incluir.

## 2. Explicar la Estructura de Su Base de Código

Comprender cómo está organizado el código permite que cualquiera saltar directamente a la acción desde el primer día. Una forma sencilla de hacerlo es **creando un mapa** de su **estructura de archivo donde explicas lo que está en él y lo que debe ir donde**. Cuando este **preste especial atención a aquellos lugares donde puede haber ambigüedad**. Por **ejemplo**, que **indica** que el **archivo "buttons.css"** **contiene estilos** para **botones** **no es muy útil**, pero que **indica** que el **directorio "personalizado"** es **donde se encuentran estilos personalizados** para el **tema puede** ser un **ahorrador de tiempo**.

Aquí está un ejemplo:

```
01 | Project Root
02 |   └─ srs
03 |       └─ styles // Base styles. Styles placed here should be available
04 |           └─ bootstrap-custom // Custom styles that overwrite bootstrap
05 |               └─ custom // Custom styles created for the application
06 |                   └─ demos // Demos to illustrate styles and interactions for the
07 |                       └─ fonts
08 |                           └─ img // Images used ONLY in stylesheets
09 |                               └─ variables // Variables used in the custom theme
10 |                                   └─ styles.less // Imports of all the base stylesheets
11 |
12 |   └─ components
13 |       └─ alerts
14 |           └─ alert.less // Custom styles for the alert component. Every compone

// Custom styles for the alert component. Every component has its own stylesheet that allows to customize it
Cada componente tiene su propia hoja de estilo que permite personalizarla específicamente para evitar la hinchazón y la pérdida de estilo.
```

Como regla general, **documentar** los **lugares** donde se **necesite aclaración**. **No cada directorio y archivo** tendrá documentación (como en el ejemplo anterior "fonts" es autoexplicativos). Ponte en los zapatos de alguien nuevo en el proyecto (o recordar

aquellos tiempos donde estabas esa persona) y proporcionar la orientación que desea le habría dado. La idea es hacerlo de una manera que no es tiempo para ti, pero es útil para evitar preguntas repetitivas.

Otro elemento clave a destacar aquí es donde se deben agregar nuevos estilos y los pasos que deben seguirse. Lo demuestra el ejemplo anterior, pero dada la naturaleza de la herencia de CSS, puede ser que vale la pena indicar en detalle.

Por ejemplo, en proyectos donde utiliza Bootstrap como un marco <sup>por debajo</sup> subyacente, por lo general tienen tres lugares donde deben ir la nuevas reglas, según lo que el desarrollador está tratando de lograr. Así que hemos añadido a una guía de la documentación que comprende tres escenarios:

## Escenario #1

Si desea sobrescribir un estilo <sup>Bootstrap</sup> definido por el sistema de arranque, entonces:

1. Averiguar en qué hoja de estilos del marco bootstrap se define la regla.
2. Ir a "src/styles/bootstrap-custom".
3. Busque la misma hoja de estilos.
4. Si no existe, crear uno nuevo en ese directorio, con el mismo nombre.
5. Añadir su sobreescritura y señalar cualquier cosa de importancia.
6. Por último, importar la hoja de estilos nuevos en "src/styles/style.less".

## Escenario #2

Si desea agregar una nueva definición de estilo que no se sobrescriban manos a la obra y debe estar disponible en cualquier lugar de la aplicación, entonces:

1. Ir a "src/styles/custom".
2. Encontrar una hoja de estilo donde se podría agregar el nuevo estilo (pensar: esto es un estilo para definir un botón, o es un estilo reutilizable como un mixin?) y donde tiene más sentido.
3. Si hay una hoja de estilo donde tiene sentido poner este nuevo estilo, a continuación, crear una nueva.
4. El nombre después de nuestra terminología.
5. Añadir su nuevo estilo y señalar cualquier cosa de importancia.
6. Por último, importar la hoja de estilos nuevos en "src/styles/style.less".

## Escenario #3

Si desea agregar una nueva definición de estilo para un componente (esto **significa** que **sólo estará disponible** para ese componente, donde el **componente se utiliza** en la **aplicación**), entonces:

1. Ir a la **"src/components"**.
2. **Encontrar** el **componente** que **desea estilo**.
3. **Encontrar** la **hoja de estilos** del **componente**, dentro del **directorio** de **componente**.
4. Por último, **añadir** el **nuevo estilo** y **señalar cualquier cosa** de **importancia**.

Esta guía:

- Sirve para **mantener** nuestros **estilos organizados**.
- **Mantener** los **estilos de trabajo** según la **herencia** que **habíamos establecido** porque sobrescribe se realizaron en los lugares correctos.
- **Evitar** los desarrolladores **escribir reglas complicadas**.
- **Evitar** **estilos escaparse** a **elementos no previstos**.

## 3. Establecer los Estándares de Codificación

La codificación de las normas o CSS guía de estilo se refiere a la forma en que su equipo ha acordado escribir CSS. Esto incluye las mejores prácticas sobre la escritura de código, como formato, **nomenclatura** y las **convenciones** de **sintaxis**. Muchas empresas han compartido la forma de hacerlo (este artículo de trucos CSS tiene una gran compilación: [Guías de Estilo CSS](#)). Aquí hay un par de maneras resulta útil para compartir este tipo de información:

### Lista de No se Debe Hacer vs. Lo Que Debe Hacer

Utilice este formato para señalar las cosas que queremos evitar, mientras que proporciona una alternativa viable. Esto elimina la ambigüedad y anima a la gente a hacer una cosa específica. Por ejemplo:

No Hacer	Hacer
<b>No</b> utilice las lengüetas para sangría. <small>Tabs</small>	Utilice <b>cuatro 4 espacios</b> por <b>indentación</b> .

No Hacer	Hacer
No utilice <code>under_scores</code> o "camelCase" <sup>para</sup> clases nombre o ID.	Utilice guiones para separar palabras.
No utilice nombres de clase y ID a reflejar la estructura subyacente de marcado. <code>.Container-span</code> y <code>.small-header-div</code> son nombres malos.	Pensar en CSS en términos de objetos y usar sustantivos simples como nombres <code>.global alert</code> y <code>.badge</code> <sup>placa</sup> son buenos nombres.
No utilice ID y selectores demasiado específicos al estilo. Usarlo sólo cuando sea absolutamente necesario (por ejemplo, controles de formulario o página de anclas). <small>form controls</small> <small>page anchor</small>	Utilizar clases para facilitar la reutilización y reducir los conflictos de especificidad CSS selector.

## Lista de Mejores Prácticas

Resumir sus pautas en las mejores prácticas e incluyen ejemplos. Esto hará que cada uno sea más fácil leer y entender. Por ejemplo:

Mejores Prácticas	Ejemplo
Escribir selectores múltiples en líneas independientes.	<code>.btn,</code> <code>.btn-link {</code> <code>}</code>
Incluir un espacio entre el selector y la llave de apertura.	<code>.Selector {</code> <code>}</code>
Use notación abreviada para valores hexadecimales cuando sea posible.	<code>#fff</code> VS <code>#ffffff</code>
Escriba valores hexadecimales en minúsculas.	<code>#3d3d3d</code> VS <code>#3D3D3D</code>
Incluir URLs en las comillas simples. En general, comillas simples se prefieren sobre comillas, ya que son fáciles de escribir.	<code>url</code> <code>(' /image.png')</code> VS <code>url ("/image.png")</code>

Mejores Prácticas	Ejemplo
No utilice las unidades para los valores cero (0), excepto para ángulos (grados) y tiempo (s o ms).	margin-right: 0; VS margin-right: 0px;

El desarrollador de una manera escribe código puede diferir mucho de otro. Por esta razón es importante para su equipo establecer estándares de codificación. Esto asegura que el código sea consistente a través de un proyecto, que hace más fácil de leer, escribir y revisar. Pero asegúrese de que cualquier cosa que incluya en sus estándares de codificación es una práctica que su equipo ha acordado.

He trabajado en un proyecto donde incluimos esto en nuestra vida guía de estilo. Como parte del código, cometidos y había empujado estas mejores prácticas para el repositorio. Para asegurarse de que todos estaban a bordo, todos en el equipo tenían que aprobar la solicitud de extracción antes de que podríamos combinar. Esto garantiza que todo el mundo tenía que hacer tiempo para revisar y discutir.

## 4. Evitar Largas Hojas de Estilo

Cuando rompes tus estilos en hojas más pequeñas y más centrado es más fácil documentarlos. También puede ahorrar tiempo al no tener que documentar lo que se hace explica por sí mismo.

Por ejemplo, en lugar de <sup>NO</sup> tener una hoja de estilos de línea 800 con todas las variables que se pueden utilizar en un tema, puede tener un archivo para cada uno de los tipos de variables. Esto ahorrará tiempo al no tener que desplazarse hacia arriba y hacia abajo en el archivo tratando de encontrar algo! Creo que así como el tiempo que ahorras al no tener que actualizar el índice cada vez que agregue o cambie el nombre de una nueva sección.

En un <sup>ARCHIVO LARGO</sup> archivo de larga, un índice de largo...

```

01  /*-----*\
02      variables.less
03
04      Index
05      - Color Palette
06      - Typography
07

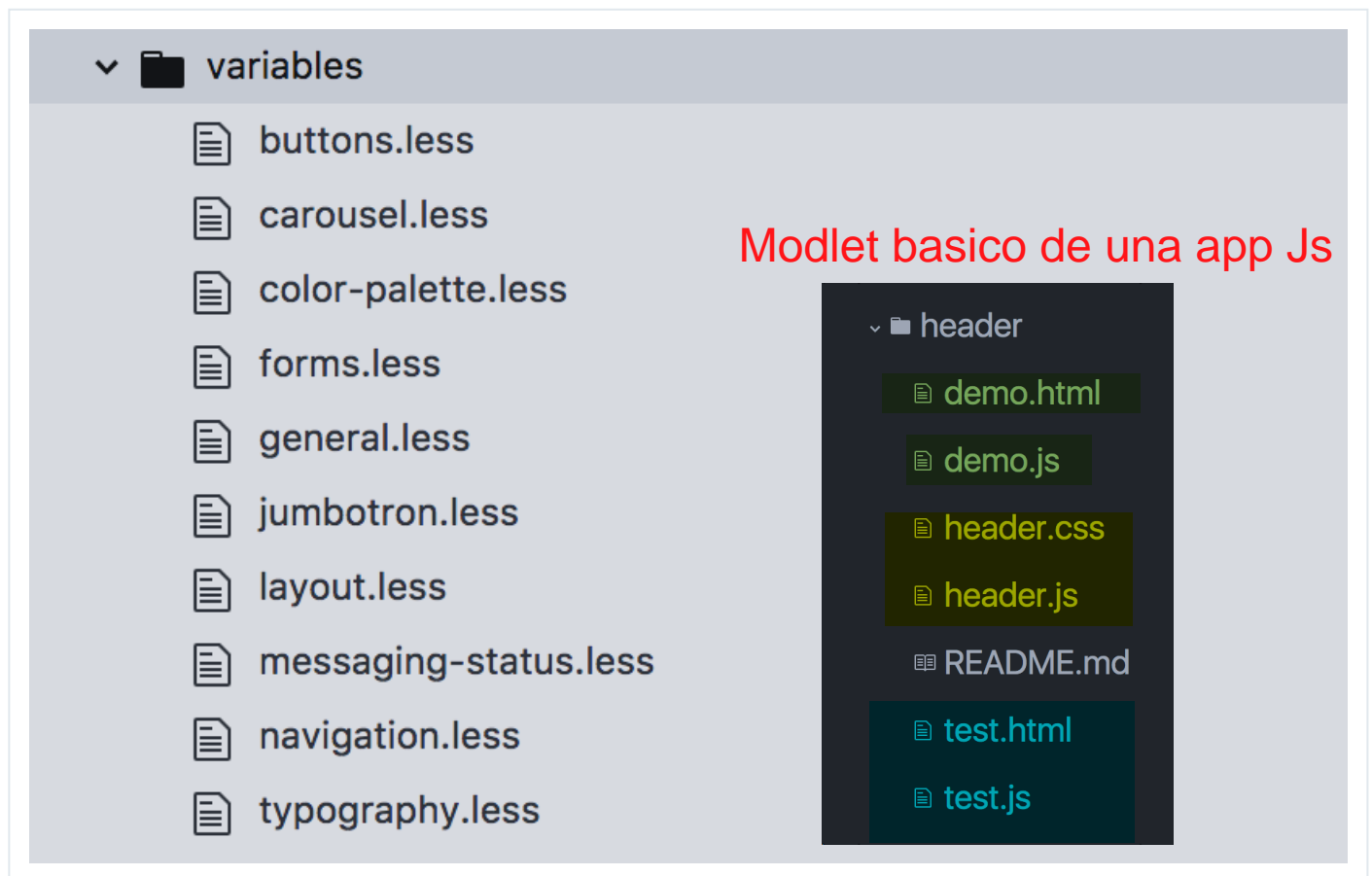
```

```

08 | - Buttons
09 | - Forms
10 | - Layout
11 | - Messaging & Status
12 | - General
13 | - Navigation
14 | - Carousel
15 | - Jumbotron
    | \*-----*/

```

Rompiendo un fichero, ningún índice es necesario:



Otro ejemplo a considerar cuando se trabaja en **grandes aplicaciones** es **el flujo de trabajo de modlet**. Dividir una App de js en pequeños modulos para poder depurarlos o actualizarlos Explica por qué trabajar con **archivos más pequeños organizado** por **componentes** **permite pruebas** y **montaje** en su **aplicación más fácilmente**.

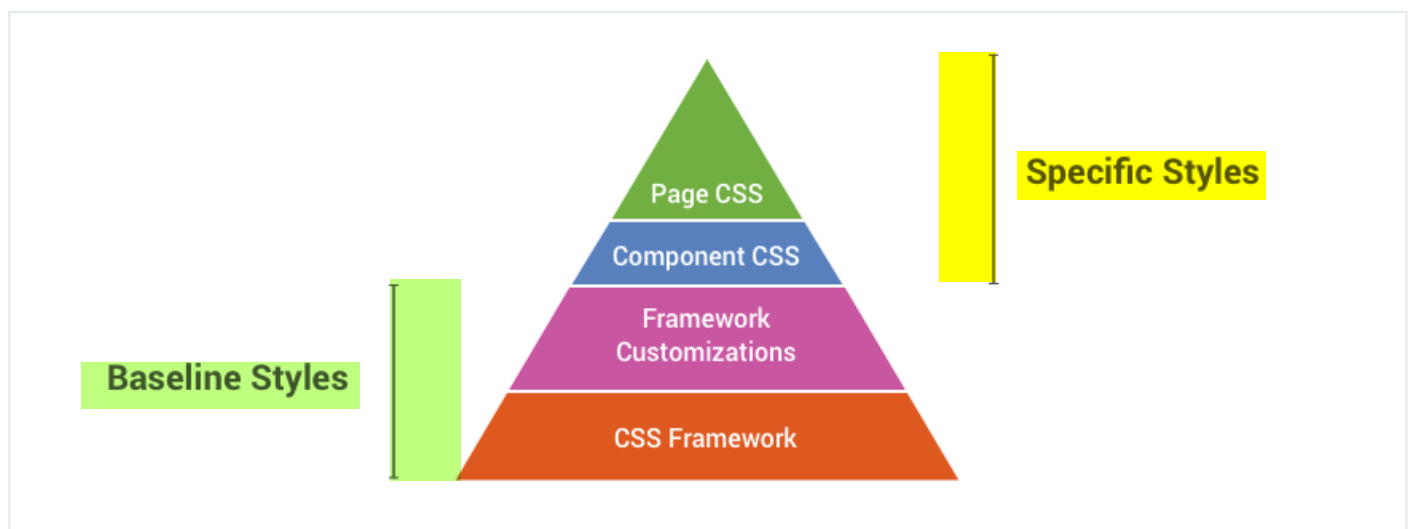
## 5. Documento CSS Con una Guía de Estilo en Mente

Una gran parte de **documentar** correctamente **CSS** tiene que ver con **escribir buena CSS** y **viceversa**. Esto significa que incluso cuando el **estado** de su **base de código** CSS

no puede ser el mejor, cumplimiento de normas de documentación puede mover hacia un sistema mejor.

Aquí es donde documentando CSS con una guía de estilo en mente entra en su lugar. La idea es que una guía de estilo puede ayudar a determinar una buena estructura para su CSS porque para crear uno necesita distinguir entre:

- los estilos de línea de base que definen la apariciencia de la aplicación (incluyendo cualquier frameworks CSS que usas)
- las personalizaciones que hacen a componentes específicos, y
- las personalizaciones que se realizan a páginas específicas.



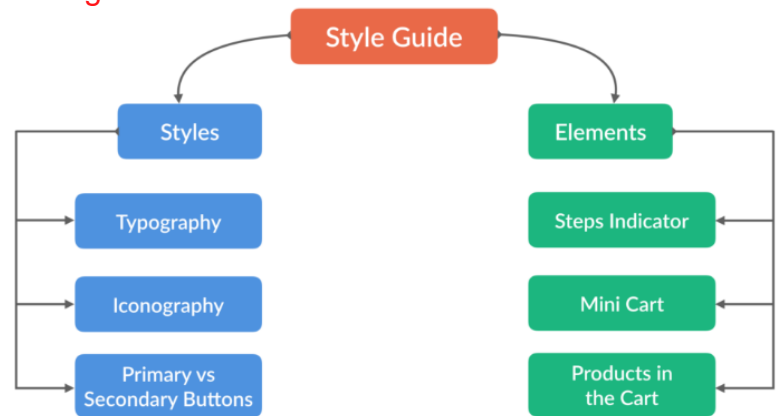
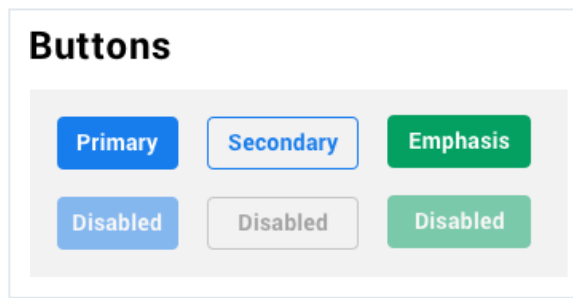
La mayor parte de su CSS debe conformada por los estilos de línea de base, ya que están disponibles en cualquier lugar de la aplicación y afectan a todos los elementos en su estado predeterminado. Estilos personalizados deben realizarse a medida que agrega los componentes con un aspecto específico y el comportamiento, o en los casos donde lo requiere el diseño de un elemento o componente en una página.

Una buena manera de captar cómo esta configuración específica puede trabajar en su aplicación <sup>grafico 1</sup> <sup>sitemap</sup> es crear un mapa de guía de estilo. Una vez que sabes cómo una guía de estilo como en su aplicación, puede documentar elementos con eso en mente. Por ejemplo, si ha definido en la guía de estilo botones y navegación el aspecto, es claro corte donde debe agregar nuevos estilos y documentación para ellos (en "buttons.css" y "navs.css"). Pero ¿qué pasa con una navegación que se hace de botones?

Tener una guía de estilo puede ayudarle a tomar esta decisión, como se puede comparar cómo ven botones y navegaciones, desde una perspectiva de marcado y una



pantalla. Veamos este ejemplo:



```

1 <button type="button" class="btn btn-primary">Primary</button>
2 <button type="button" class="btn btn-secondary">Secondary</button>
3 <button type="button" class="btn btn-emphasis">Emphasis</button>
4
5 <button type="button" class="btn btn-primary disabled">Primary</button>
6 <button type="button" class="btn btn-secondary disabled">Secondary</button>
7 <button type="button" class="btn btn-emphasis disabled">Emphasis</button>

```

## Navigation

Active Nav Item Nav Item Disabled

```

1 <ul class="nav nav-tabs">
2   <li class="active"><a href="#">Active</a></li>
3   <li><a href="#">Nav Item</a></li>
4   <li><a href="#">Nav Item</a></li>
5   <li class="disabled"><a href="#">Nav Item</a></li>
6 </ul>

```


En este caso, existen dos posibles ubicaciones para el CSS que definirá la navegación de botones:

1. Si la marca sigue la estructura de otras navegaciones, utilizando una lista de enlaces, o un `<nav>` con enlaces que parecen botones, luego añadir los estilos nav a "navs.css".
2. Si el marcado que se utiliza es `<button>` luego, añadir los estilos a "buttons.css". Incluso se podría añadir como una hoja de estilo separada (como "botones-groups.css" group.css"). En este caso, el término "navegación" no sería apropiado más botones HTML son menos accesibles como elementos de navegación.

## 6. Rompa Sus Hojas de Estilos en Secciones

Una vez que se ha venido abajo sus **hojas de estilos** en **archivos más manejables**, puede **continuar** este ejercicio por **romper cada estilo** en **secciones individuales**.

Cada hoja debe incluir al menos un título y (útil) una breve descripción. El título podría ser tan simple como el nombre del archivo, **capitalizado** para se parecen más a un título (ex: "Botones" de la hoja de estilos "buttons.css"), o podría ser el mismo que el nombre del archivo, así:



Botones

```
01  /**
02   * icons.css
03   */
04
05  .icon {
06    font-family: 'bitovi';
07    speak: none;
08    font-style: normal;
09    font-weight: normal;
10    font-variant: normal;
11    text-transform: none;
12    line-height: 1;
13  }
```

Encontrar usando el **nombre de archivo** especialmente útil al **depurar el código** en el **navegador**, y sobre todo cuando el **archivo** ha sido **compilado con otros archivos**, como puedo conseguir una referencia al archivo donde vive el estilo.

También, tenga en cuenta que el estilo de comentario que se abre con  `/ **` vs solo  `/ *`. Esto es una Convención utilizada en **JSDoc** para analizar los comentarios que deben incluirse en la documentación generada automáticamente. Recomiendo usar este estilo, tantas vida uso de generadores de guía de estilo del JSDoc formato, cuando estés listo para usar un generador, el código va a necesitar muy poco trabajo adicional.

En cualquier caso, puede **utilizar otros estilos** para **denotar una sección** tales como:

```
01  /*-----*\
02   icons.css
03  \*-----*/
04
05  .icon {
06    font-family: 'bitovi';
07    speak: none;
08    font-style: normal;
09    font-weight: normal;
10    font-variant: normal;
11    text-transform: none;
12    line-height: 1;
13  }
```

Hasta cierto punto, esto depende de qué equipo está de acuerdo es la mejor manera de hacer una sección que destaca. El único requisito es utilizar `/ *` al principio y `* /` al final. Lo que realmente importa es que cualquiera que te utiliza, palillo a él y utilizarlo a través de tu código CSS de una manera consistente.

Si usted piensa que una descripción puede ser útil en una determinada hoja de estilo, luego añadirlo como parte de este primer comentario. Por ejemplo:

```
01  /**
02   * icons.css
03   * 80 caracteres como max
04   * Icons should convey in a simple and meaningful way the concept of the function
05   * they represent. When designing new icons be sure to remove any complexities
06   * and follow the linear and lightweight appearance of the icon set.
07   */
08
09  .icon {
10    font-family: 'bitovi';
11    speak: none;
12    font-style: normal;
13    font-weight: normal;
14    font-variant: normal;
15    text-transform: none;
16    line-height: 1;
17  }
```

Los iconos deben transmitir de manera simple y significativa el concepto de la función que representan. Al diseñar nuevos íconos, asegúrese de eliminar cualquier complejidad y siga el aspecto lineal y liviano del conjunto de íconos.

Esto reforzará la idea de que una sección. Asimismo, tratar de romper la descripción en varias líneas (Harry Roberts recomienda hasta 80 caracteres) para que sea más fácil de leer mientras que teniendo varios archivos abiertos o al leer en Github.

Después de agregar un título y una descripción, puede ir un paso más por romper los estilos dentro de la hoja de estilos en secciones. Para hacer este pensar en cómo lógicamente explicando el contenido de una hoja de estilo tiene sentido. Por ejemplo, la hoja de estilos "buttons.css" por lo general tendrá una <sup>comentario</sup> sección donde se define el estilo por defecto de un botón aplicando sólo el `.btn` clase. Entonces habrá más estilos que definen diferentes colores, tamaños y configuraciones que se pueden aplicar conjuntamente para personalizar su aspecto y comportamiento. Crear secciones para cada uno de esos estilos hará más fácil entender y encontrar donde nuevas clases o sobrescribe debe aparecer. También, es menos intimidante para consultar un archivo cuando el código se presenta en fragmentos versus un archivo largo que es difícil decir donde estilos comienzan y terminan.

Echemos un vistazo a este ejemplo comparativo. Primero, un bloque de código LESS sin las secciones:

```
01 .label-condensed-variant(@color) {
02   &:before {
03     background-color: @color;
04   }
05 }
06
07 .label {
08   border-radius: 0;
09   font-family: @font-family-bold;
10   font-size: 85%;
11   position: relative;
12   &.label--condensed {
13     font-size: @font-size-xsmall;
14     color: @gray;
15     background: transparent;
16     padding-right: @padding-small;
17     &.label--primary {
18       .label-condensed-variant(@label-primary-bg);
19     }
20     &.label--success {
21       .label-condensed-variant(@label-success-bg);
22     }
23     &.label--info {
24       .label-condensed-variant(@label-info-bg);
25     }
26     &.label--warning {
27       .label--condensed-variant(@label-warning-bg);
28     }
29     &.label--danger {
30       .label-condensed-variant(@label-danger-bg);
31     }
32   }
33   &.label--simple {
34     font-family: @font-family-base;
35     font-size: @font-size-xsmall - 1;
36     color: @gray;
37     border: 1px solid @gray-light;
38     padding: 2px;
39     border-radius: @border-radius-small;
40     text-transform: uppercase;
41   }
42 }
```

Y el mismo bloque de código con las secciones:

```
01 /**
02  * bootstrap-custom/_labels.less Labels
03  *
04  * Overwrites the default styles defined by the bootstrap framework.
05  *
06  */
07
08 .label {
09
```

Seccion

Section

```
10 border-radius: 0;
11 font-family: @font-family-bold;
12 font-size: 85%;
13 position: relative;
14 }
15
16
17 /**
18  * Condensed Labels
19  *
20  * Modifies labels to provide a smaller and narrower version with a colored circle t
21  */
22
23 .label {
24   &.label--condensed {
25     font-size: @font-size-xsmall;
26     color: @gray;
27     background: transparent;
28     padding-right: @padding-small;
29   }
30 }
31
32
33 /**
34  * Condensed Labels - Colors
35  */
36
37 .label-condensed-variant(@color) { // Variant mixin to set the circle color
38   &:before {
39     background-color: @color;
40   }
41 }
42
43 .label {
44   &.label--condensed {
45     &.label--primary {
46       .label-condensed-variant(@label-primary-bg);
47     }
48     &.label--success {
49       .label-condensed-variant(@label-success-bg);
50     }
51     &.label--info {
52       .label-condensed-variant(@label-info-bg);
53     }
54     &.label--warning {
55       .label-condensed-variant(@label-warning-bg);
56     }
57     &.label--danger {
58       .label-condensed-variant(@label-danger-bg);
59     }
60   }
61 }
62
63
64 /**
65  * Simple Labels
66  *
67  * Modifies labels to provide a simple linear version where colors are not used.
68  */
69
70 .label {
```

Section

```
71 | &.label--simple {
72 |   font-family: @font-family-base;
73 |   font-size: @font-size-xsmall - 1;
74 |   color: @gray;
75 |   border: 1px solid @gray-light;
76 |   padding: @padding-small;
77 |   border-radius: @border-radius-small;
78 |   text-transform: uppercase;
79 | }
```

## 7. Índice del Contenido de Sus Hojas de Estilos

Esto es una gran manera de proporcionar una instantánea de lo que es en la hoja de estilos y una necesidad en aquellos proyectos donde, por cualquier razón, largas hojas de estilo están ahí para quedarse (no un ventilador de esos proyectos, pero suceden!).

Un **índice** de la **hoja** de **estilos** normalmente se ve así:

```
01 | /**
02 |  * icons.css
03 |  *
04 |  * Icons should convey in a simple and meaningful way the concept of the function
05 |  * they represent. When designing new icons be sure to remove any complexities
06 |  * and follow the linear and lightweight appearance of the icon set.
07 |  *
08 |  * Index
09 |  * - Icon Font
10 |  * - Icon Variations
11 |  * - Icon Animations
12 |  *
13 | */
```


Y aunque me encanta cómo aseada y útil pueden ser, tengo que admitir que pueden ser fácilmente olvidados y anticuados por lo tanto. También son un dolor para actualizar cuando son largas y utiliza números (para evitar aquellos!)

Una manera alternativa de utilizar índices es dejar un estilo de guía generador el trabajo por usted al mirar sus hojas de estilos, encontrar las secciones han definido y generar un índice para usted. Me extenderé más sobre este tema al final de este artículo.

## 8. Encontrar el Punto Dulce de Documentar


Aquí radica el secreto de la documentación. Es fácil dejarse llevar y entrar en un frenesí de documentación una vez, para luego olvidarse de ella y terminan con solamente una porción de su codebase demasiado documentados y los indocumentados del resto. Como con todo en la vida, el secreto es encontrar el equilibrio. Aquellas **áreas** donde la **atención** es necesaria porque hay **imprevistas dependencias**, **recursos adicionales** o **notas importantes** a tener en cuenta el **documento**. Es decir que no cada pedacito de código debe ser documentada pero es definitivamente útil **romper trozos** y **explicar cuáles** son **esos trozos** cuando **sea necesario**. De esta manera, documentación se convierte en una útil herramienta que es parte de su flujo de trabajo y no una ocurrencia tardía que evitan hacerlo. Pero, exactamente ¿cómo? Aquí está un ejemplo:

Digamos que vas a aplicar el marcado y el CSS para el siguiente componente de la tarjeta:



Purple Dalia

Curabitur non leo quis magna vulputate dapibus eget at turpis. Vivamus nec metus at mi laoreet eleifend et quis nisi. Interdum et malesuada fames ac ante ipsum primis in faucibus. Suspendisse malesuada viverra nulla

 Article

`.card_image`

`.card_content`

`.card_footer`





## Purple Dahlia

Dahlias are annual blooming plants, with mostly tuberous roots. While some have herbaceous stems, others have stems which lignify in the absence of secondary tissue and resprout



Article



## Daffodil

Daffodils are scapose, having a single central leafless hollow flower stem (scape). Several green or blue-green, narrow, strap-shaped leaves arise from the bulb. The plant stem usually bears a



Podcast



## Poppies

A poppy is a flowering plant in the subfamily Papaveroideae of the family Papaveraceae. Poppies are herbaceous plants, often grown for their colourful flowers. One species of poppy, *Papaver*



Article



## Frangipan

Frangipania are flowers are most fragrant at night in order to lure sphinx moths to pollinate them. The flowers yield no nectar, however, and simply trick their pollinators. The moths



Video



## Blue Marquerite

The blue marguerite or blue daisy, is a species of flowering plant of the family Asteraceae, native to South Africa. *F. amelloides* is synonymous with, and formerly known as, *F. aethiopica*, *Aster*



Article



## Gerbera Jamesonii

*Gerbera jamesonii* is a species of flowering plant in the genus *Gerbera*. It is indigenous to South Eastern Africa and commonly known as the Barberton daisy, the Transvaal daisy, and as



Article



[Homemade organic fertilizers recipes](#)



[Using chickens in your garden](#)



[Why mulching is so important in the winter?](#)



[When to pruner and fertilize roses](#)





## Purple Dahlia

Dahlias are annual blooming plants, with mostly tuberous roots. While some have herbaceous stems, others have stems which lignify in the absence of secondary tissue and resprout following winter dormancy, allowing...



Article



## Daffodil

Daffodils are scapose, having a single central leafless hollow flower stem (scape). Several green or blue-green, narrow, strap-shaped leaves arise from the bulb. The plant stem usually bears a solitary flower, but occasional...



Podcast

Mirando los diseños puede identificar los siguientes patrones de estilo:

- El diseño de la base tarjeta
- La red de tarjetas
- La lista de tarjetas
- La versión oscura de tarjeta

The card base design

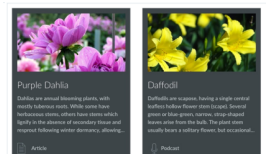
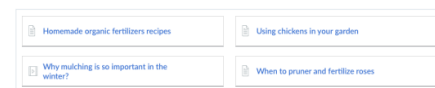
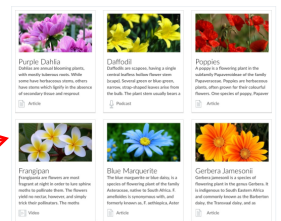
The cards grid

The cards list

The card dark version

Puede entonces descomponer a la implementación de CSS con los patrones en la mente y utilizar la documentación que le guiará. Para comenzar con tu hoja de estilo "cards.css" puede incluir una introducción simple como sigue:

```
01  /**
02   * cards.css
03   *
04   * These are the styles for the card component.
05   *
06   * Index
07   * - Card Base
08   * - Card Grid
09   * - Card List
10   * - Card Dark
11  */
```



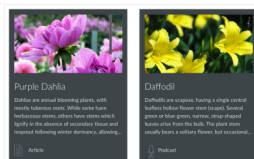
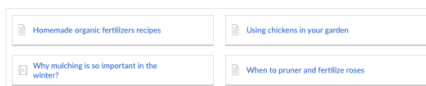
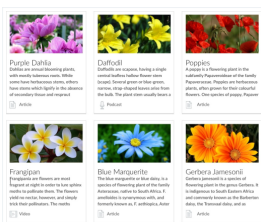
Puede incluir información más útil en la introducción, pero ya que es sólo introducción algo sencillo puede ayudar a establecer el esqueleto de la documentación.

Entonces, vamos a agregar las secciones donde usted va a trabajar sus estilos en:

```

01  /**
02  * cards.css
03  *
04  * These are the styles for the card component.
05  *
06  * Index
07  * - Card Base
08  * - Card Grid
09  * - Card List
10  * - Card Dark
11  */
12
13
14  /**
15  * Card Base
16  */
17
18
19  /**
20  * Card Grid
21  */
22
23
24  /**
25  * Card List
26  */
27
28
29  /**
30  * Card Dark
31  */

```



Con estas secciones en mente, se pueden **visualizar** cómo el **código debe ser estructurado**. Usted sabe que usted debe hacer las definiciones de **base** de la **tarjeta flexible** y suficientemente **independiente** que usted puede **hacer** fácilmente la **tarjeta de trabajo** en una **cuadrícula**, **lista** o **versiones oscuras**.

Entonces como se escribe el código, usted puede conseguir más específica con tus comentarios:

```

01  /**
02  * Card Base
03  *
04  * Defines the appearance and behaviour of the default card with its main parts:
05  * - Card Image
06  * - Card Content
07  * - Card Footer
08  */
09
10  .card {...}
11  .card_image {...}
12  .card_logo {...}
13  .card_content {...}
14  .card_footer {...}

```



Considero el **nivel básico** de **documentación** que debe **incluir** porque **sirve como guía** **para** el **diseño del código** y rápidamente **informa cómo se organizan** las **cosas** para la siguiente persona que trabaja en él.

El **siguiente nivel** es **agregar comentarios** que son **específicos** a una **regla**, y que puede **utilizarse** para **explicar** lo que está **haciendo esta regla** porque no es obvio a simple vista. Por ejemplo:

```
01  /**
02   * Card Base
03   *
04   * Defines the appearance and behaviour of the default card with its multiple parts:
05   * - Card Image
06   * - Card Logo
07   * - Card Content
08   * - Card Footer
09  */
10
11  .card {
12    @media (max-width: {{ screen-xs }} ) {
13      border: none; /* because the card takes 100% of the screen on mobile */
14    }
15  }
16  .card__image {...}
17  .card__logo {...}
18  .card__content {
19    flex: 1 1 auto; /* "auto" needs to be explicit to avoid the div collapsing on IE
20  }
```

La belleza de este enfoque es que la documentación está allí para apoyar e informar a la aplicación como vas, frente a algo que se añade en un momento posterior.

Aquí hay un par de ejemplos más de **framework Bootstrap** que muestran cuando los comentarios son útiles y cuándo vale la pena entrar en más detalles.

## Ejemplo #1

```
1  // Need .dropdown-toggle since :last-child doesn't apply,
2  // given that a .dropdown-menu is used immediately after it
3
4  .btn-group > .btn:last-child:not(:first-child),
5  .btn-group > .dropdown-toggle:not(:first-child) {
6    .border-left-radius(0);
7  }
```

Este comentario aclara por qué estos **estilos** existen y lo que están haciendo. También es corta y al punto, comunicando la idea en un lenguaje informal.

## Ejemplo #2

```
01 // Checkbox and radio options
02 //
03 // In order to support the browser's form validation feedback, powered by the
04 // `required` attribute, we have to "hide" the inputs via `clip`. We cannot use
05 // `display: none;` or `visibility: hidden;` as that also hides the popover.
06 // Simply visually hiding the inputs via `opacity` would leave them clickable in
07 // certain cases which is prevented by using `clip` and `pointer-events`.
08 // This way, we ensure a DOM element is visible to position the popover from.
09 //
10 // See https://github.com/twbs/bootstrap/pull/12794 and
11 // https://github.com/twbs/bootstrap/pull/14559 for more information.
12
13 [data-toggle="buttons"] {
14   > .btn,
15   > .btn-group > .btn {
16     input[type="radio"],
17     input[type="checkbox"] {
18       position: absolute;
19       clip: rect(0,0,0,0);
20       pointer-events: none;
21     }
22   }
23 }
```

Esto es un gran ejemplo de la documentación que va más en profundidad, explicando la lógica detrás de una decisión de implementación y proporciona vínculos a información adicional.

## Tomando un Paso Más

Con estas mejores prácticas en la mente, el siguiente paso es incorporar *vida guía de estilo* como parte de su documentación. Vida guía de estilo es un documento vivo que muestra los comentarios que han incluido en el código estructurado como un sitio web, para que puedas navegar la documentación por separado del código fuente.

Lo que hace vivir guías de estilo de gran alcance es que la documentación actual vive con el código y puede ser fácilmente actualizado como los cambios de código, permitiendo permanecer en sincronización y relevante. La ventaja adicional es que usted puede hacer esta documentación disponible a otras personas en tu equipo que no puede interactuar directamente con el código (como diseñadores, gerentes de producto, ingenieros de control de calidad). Estos miembros del equipo también resultaría útil saber cómo se perfila la interfaz de usuario.

En la vida de una guía de estilo, puede incluir demostraciones interactivas de su código y además pueden organizar la documentación independientemente de la estructura del código.

## Conclusión

Documentación de CSS comienza con reglas claras y una base de código bien estructurado. Cuando se hace como parte de su flujo de trabajo, también puede servir como una guía para estructurar el código y mantenerlo organizado como crece. Esto tiene el beneficio adicional de hacer claro donde las cosas son y donde debe agregar nuevo código, facilitando la inducción de nuevos miembros al mismo tiempo de acelerado desarrollo.

## Recursos Utiles

- [styleguidedrivendevelopment.com](https://styleguidedrivendevelopment.com)



### DOCUMENTACIÓN

#### Documentar Sus Proyectos Con Daux.io

Daniel Pataki



### CSS

#### Comprensión CSS Estadísticas: Cómo Aprovechar al Máximo de los Números

Jonathan Cutrell

## Recordatorio: 8 Buenas Prácticas

1. Establecer las reglas
2. Encontrar el punto dulce de documentar
3. Índice el contenido de sus hojas de estilos

4. Desglose sus hojas de estilos en las secciones
5. Documento CSS con una guía de estilo en mente
6. Evitar largas hojas de estilo
7. Establecer sus estándares de codificación
8. Explicar la estructura de su base de código

Advertisement



**Adriana De La Cuadra**

Austin, TX

I'm a UX designer at Bitovi, a Design and JavaScript development consulting company in Chicago that authors and maintains several open source JavaScript front-end frameworks, including CanJS and DoneJS. Through my consulting work, I've had the opportunity to work with Fortune 500 Companies, working in the design of large e-commerce sites and complex internal applications. I like to geek about documentation and homemade bread. You can read more of my design articles on the Bitovi blog.

 [adrifolio](#)

## Weekly email summary

Subscribe below and we'll send you a weekly email summary of all new Web Design tutorials. Never miss out on learning about the next big thing.

Update me weekly

Advertisement

### Translations

Envato Tuts+ tutorials are translated into other languages by our community members—you can be involved too!

Translate this post

Powered by



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Matias Araus

• 5 months ago

excelente!!

^

 | 

^

 • Reply • Share >

amazon

AbelardoLG

• 6 months ago

Muy buen artículo, la pena es que la traducción evita que sea fácil de digerir.

^

 | 

^

 • Reply • Share >

Advertisement





tuts+

28,042  
Tutorials

1,260  
Courses

39,815  
Translations

---

[Envato.com](#) [Our products](#) [Careers](#) [Sitemap](#)

© 2019 Envato Pty Ltd. Trademarks and brands are the property of their respective owners.

Follow Envato Tuts+     
[Facebook](#) [Twitter](#) [Pinterest](#)