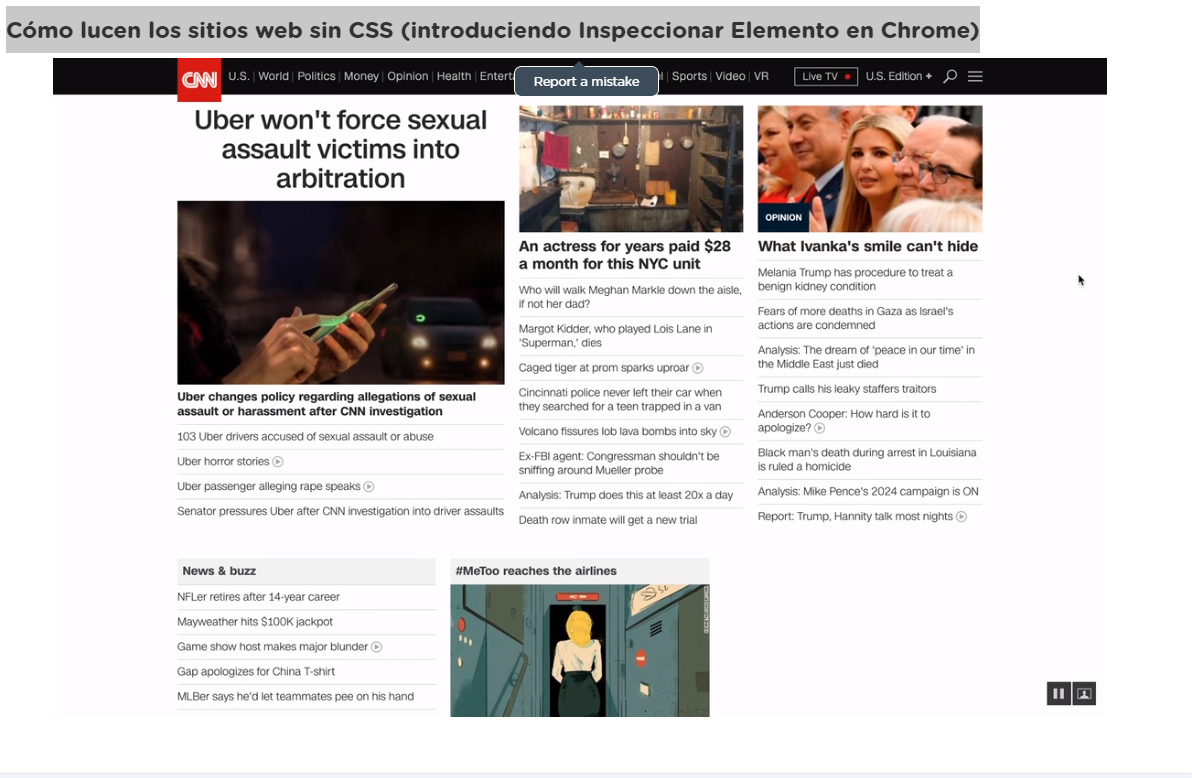
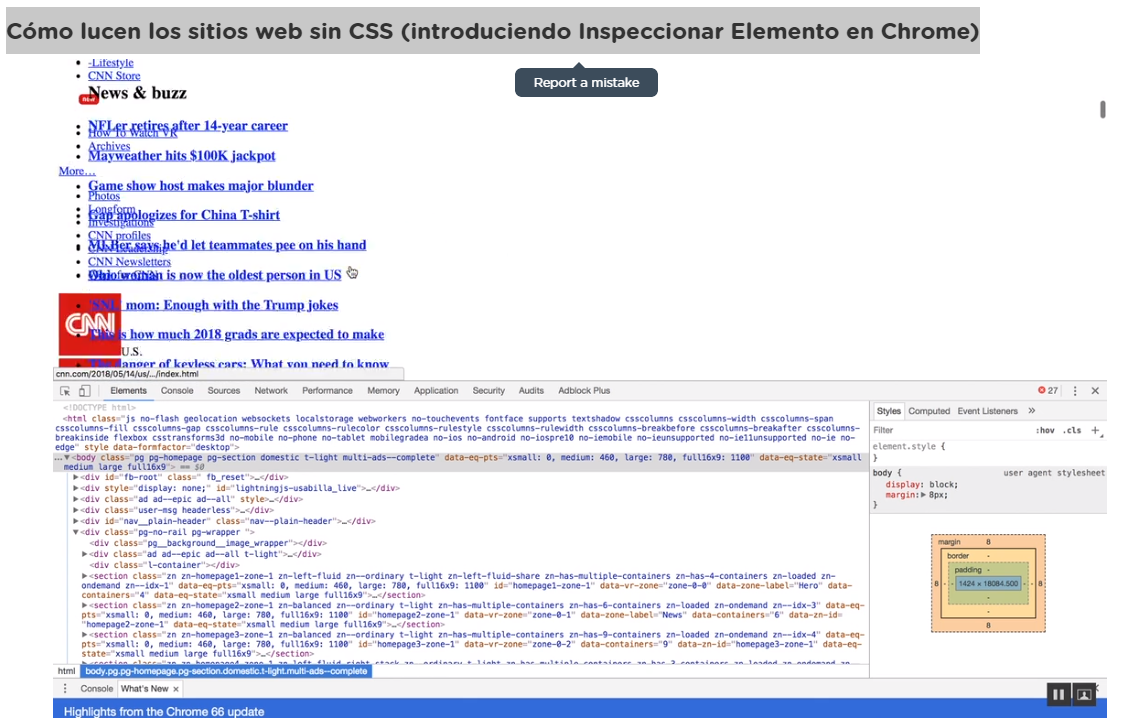
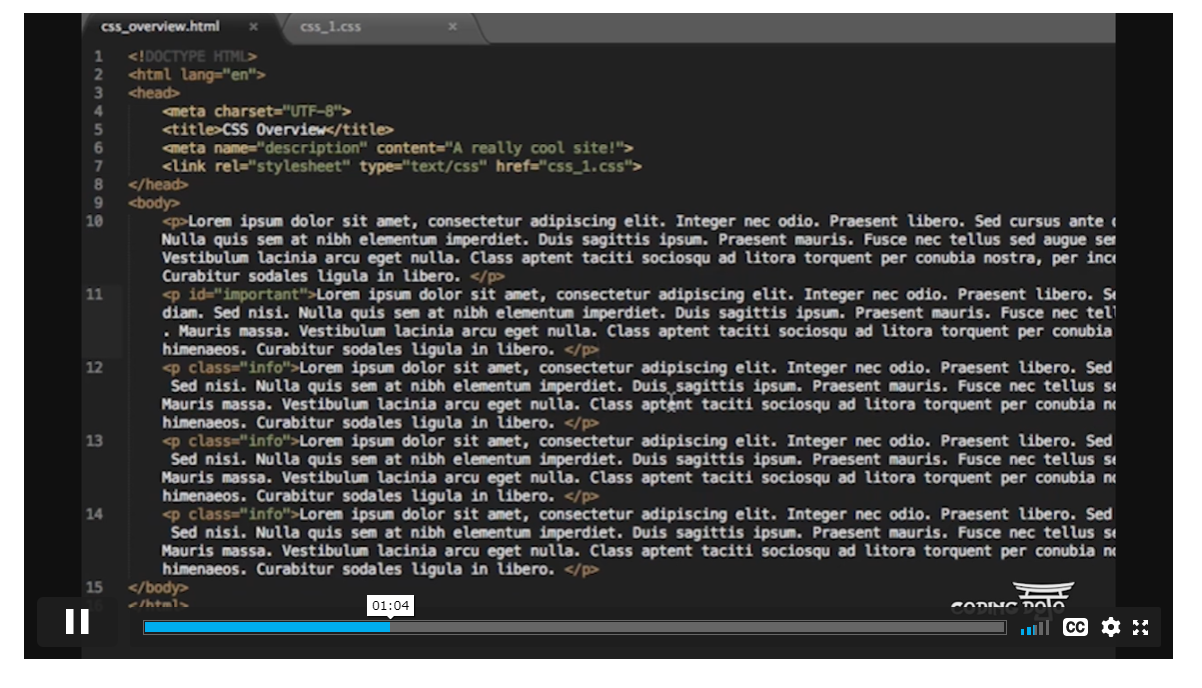
**Introducción**

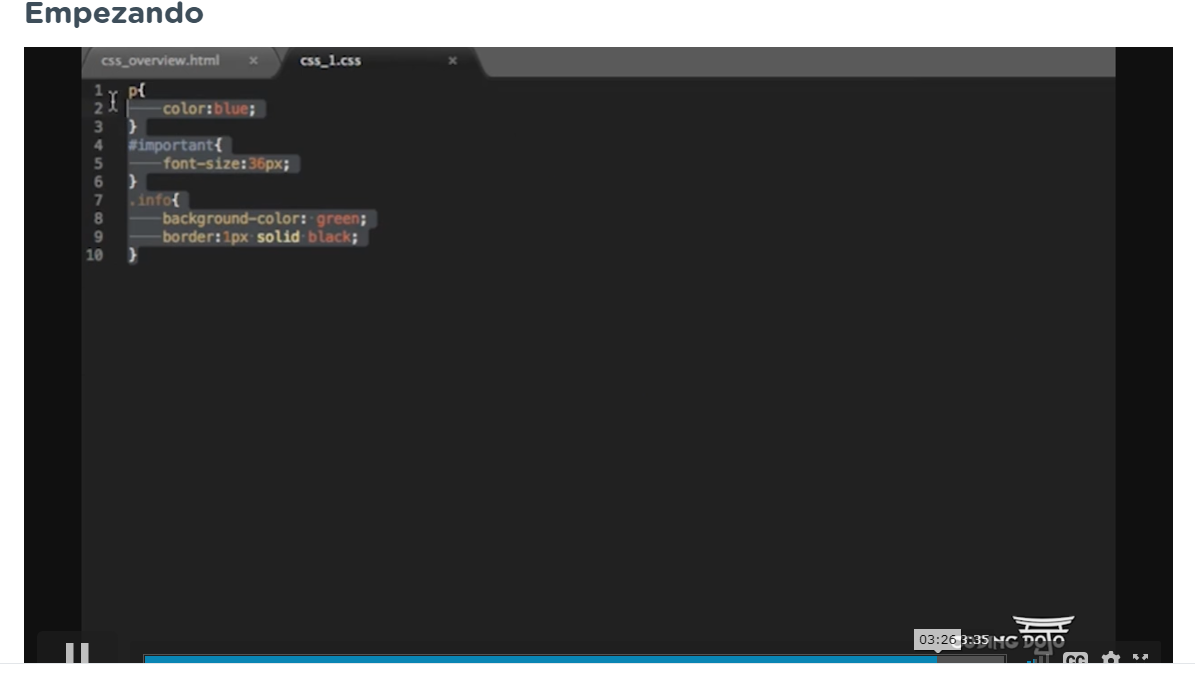
**Cómo lucen los sitios web sin CSS (introduciendo Inspeccionar Elemento en Chrome)**





# Empezando





\*Puedes descargar el código de ejemplo del video anterior [acá (css-overview.zip)](http://s3.amazonaws.com/General_V88/boomyeah/company_209/chapter_3016/handouts/chapter3016_4894_css-overview.zip).

Hay 3 formas de adjuntar CSS a un documento:  **inline, internal, y external.** **Inline e Internal son consideradas malas prácticas.**Solo utilizaremos external stylesheets, enlazadas desde el documento HTML utilizando la etiqueta <head>. Revisa el curso de HTML si no recuerdas cómo hacer esto.

CSS utiliza un esquema de prioridad para determinar qué reglas de estilo aplicar si más de una regla encaja con un elemento particular. En este esquema, las prioridades o pesos ("weights") bajan en cascada y son asignadas a reglas, por lo que los resultados son predecibles.

Una hoja de estilo es una lista de **reglas** (rules). Cada regla (o “rule-set”) consiste en uno o más **selectores** (selectors) y un **bloque de declaración** (declaration block).

# CSS Selector

Los selectores se utilizan para declarar los elementos de HTML para los cuales aplicará un estilo.

Los selectores más comunes son:

* Todos los elementos con una etiqueta específica de HTML (ej: p, h1)
* Los elementos especificados de acuerdo a los siguientes atributos:
  + **id**(son precedidos por # en CSS)
    - Debe ser único y solo puede ser utilizado una vez en tu página.
  + **class** (son precedidos por **.** en CSS)
    - Pueden ser utilizados muchas veces para compartir código CSS repetido.

#### **Bloque de Declaración**

Un bloque de declaración es una lista de declaraciones entre llaves “{}”. Cada declaración individual consiste en una **propiedad**, dos puntos ( : ), y un **valor**. Si un bloque tiene múltiples declaraciones, tienen que estar separadas por un punto y coma ( ; ).

Algunos ejemplos:

Para todas las etiquetas “p” de HTML, haz el color de fuente azul (blue).

p {

color: blue;

}

Para un elemento con el id “importante”, haz el tamaño de fuente (font size) de 36px.

#importante {

font-size: 36px;

}

Para todos los elementos con la clase “info”, haz el fondo (background) verde y agrega un borde negro (black border) de 1px de ancho.

.info {

background-color: green;

border: 1px solid black;

}

En estos ejemplos, **p**, **#importante** e **.info** son **selectores**. **Color, font-size, background-color** y **border** son **propiedades**.

**Más sobre selectores:**

* <http://www.webdesignerdepot.com/2013/08/10-css-selectors-you-shouldnt-code-without/>
* <http://www.w3schools.com/cssref/css_selectors.asp>

Ahora que sabemos un poco más sobre cómo usar selectores, hagamos una lista general de las propiedades internal para CSS:

1. element (1 punto)
2. .class (10 puntos)
3. #id (100 puntos)

Este es el orden de prioridad predeterminado. Adicionalmente, la especificidad también cuenta, por ejemplo, ul li anulará a li. W3C ha creado un buena tabla respecto a cómo deberías calcular el peso interno (internal weight):

li {...} /\* a=0 b=0 c=1 -> specificity = 1 \*/

ul li {...} /\* a=0 b=0 c=2 -> specificity = 2 \*/

ul ol li {...} /\* a=0 b=0 c=3 -> specificity = 3 \*/

li.red {...} /\* a=0 b=1 c=1 -> specificity = 11 \*/

ul ol li.red {...} /\* a=0 b=1 c=3 -> specificity = 13 \*/

#list {...} /\* a=1 b=0 c=0 -> specificity = 100 \*/

* **a** representa el número de atributos #id en el selector
* **b** representa el número de atributos de clase (class)
* **c** representa el número de nombres de etiqueta

Combinar esto en un número te dará el peso real. Esto significa que li #list tendrá el mismo peso que ul #list. Esto es, probablemente, una de las cosas más confusas del esquema en cascada (cascade scheme), pero es de hecho más simple de lo que piensas: **se trata de contar.**

# Elementos de Estilo

Más abajo se presentan algunas de las propiedades de estilo más comunes que afectarán a todos los elementos. No es una lista definitiva y debieras buscar otras propiedades que puedan ayudarte a agregar estilo a tus documentos.

### width | height:

Las propiedades width (ancho) y height (altura) son utilizadas para determinar el tamaño de tus elementos. Los valores pueden ser expresados en pixeles (px) y porcentaje (%). Cuando se trabaja con contenido estático, es recomendable utilizar px puesto que definirás tu página para que no cambie. Si trabajas con un diseño responsivo, es mejor utilizar %,

Ten cuidado al establecer tus propiedades de altura ya que esto determinará cuánto contenido puede tener el elemento. Si no lo configuras, tu elemento se extenderá para encajar con el contenido. Por otro lado, si lo configuras y tienes más contenido que el que se puede mostrar, entonces tendrás que ajustar la altura manualmente cada vez que cambies el contenido o deberás utilizar la propiedad overflow.

Cuando quieras redimensionar tu elemento (ej: imágenes), puedes ajustar una propiedad (ancho o altura) y la otra se ajustará automáticamente para conservar las dimensiones relativas de la imagen.

a {

width: 25px;

}

div {

width: 100%;

height: 200px;

}

img {

width: 250px;

}

### overflow:

Esta propiedad determina qué debe pasar cuando el contenido dentro de un contenedor es demasiado para el tamaño del contenedor. Puedes configurar el contenedor para ocultar la información adicional que no cabe, mostrar la información como sea o agregar una barra de desplazamiento (scroll bar) con lo que el contenido podría ser visto pese a su tamaño.

div {

overflow: scroll;

}

### background:

La propiedad background (fondo) puede modificar el fondo de un elemento en una sola línea, lo que es más sencillo que separar cada propiedad en su propia línea de código. El color puede ser definido utilizando hex, rgb o código semántico (semantic code).

p {

background: #ffffff url("cherries.png") no-repeat fixed center;

}

### background-color | background-image | background-position | background-size | background-repeat:

Estas propiedades ajustan el fondo de acuerdo a su tipo. Como background, background-color puede ser definida utilizando hex, rgb o código semántico.

p {

background-color: blue;

}

div {

background-image: url("cherries.png");

background-position: center;

background-size: auto;

background-repeat: no-repeat;

}

### border:

Esta propiedad ajusta todos los elementos de borde en una sola línea. Así, el primer valor es el grosor del borde. El segundo valor es tipo de borde, mientras que el tercero es el color del borde. El color puede ser hex, rgb o código semántico.

La propiedad border puede separarse en distintas líneas utilizando border-width, border-style y border color. Además, puedes seleccionar específicamente qué borde modificar utilizando border-top (arriba), border-bottom (abajo), border-right (a la derecha) y border-left (a la izquierda).

button {

border: 2px dotted green;

}

div {

border: 1px solid #000000;

}

p {

border-right: 1px groove rgb(100,100,100);

border-left: 1px groove rgb(200,200,200);

}

### border-radius

Esta propiedad permite dar una apariencia curva a las esquinas de tu borde. Los valores puede ser establecidos utilizando px o %.

button {

border-radius: 5px;

}

# Texto de Estilo

A continuación algunas de las más comúnmente utilizadas propiedades de estilo de texto (text style properties). Esta no es la lista completa, por lo que te invitamos a buscar en internet otras propiedades de estilo de texto..

### color:

El color de texto se especifica usando la propiedad de color. El valor de la propiedad de color puede ser expresada utilizando hex, RGB o código semántico.

p {

color: #ffffff;

}

a {

color: rgb(255, 255, 255);

}

span {

color: white;

}

### text-align:

Esta propiedad se utiliza para establecer la alineación horizontal de cualquier texto. El texto puede estar centrado, alineado a la izquierda o a la derecha, o justificado. Esta propiedad solo funcionará si también presenta un bloque.

h1 {

text-align: center;

}

### text-decoration:

Es utilizada para agregar o remover underlines (subrayados), overlines (línea sobre) y line-throughs (tachado).

a {

text-decoration: none;

}

## Font (Fuente)

### font-family:

La propiedad font-family especifica el estilo de fuente a ser usado por un elemento. Existen 2 tipos de nombres de font-family:

* named-family - Ejemplos: "times", "courier", "arial"
* generic-family - Ejemplos: "serif", "sans-serif", "cursive", "fantasy", "monospace"

**Fuentes Web-safe:**

Existen fuentes instaladas en casi todos los sistemas y se denominan web safe fonts (fuentes seguras para la web), porque en general se verán igual independiente del sistema que tenga el usuario. Algunas de ellas son: Verdana, Arial, Trebuchet MS, Times New Roman, Georgia, Andale Mono, Courier New, Comic Sans e Impact.

p {

font-family: "helvetica neue", arial, verdana, sans-serif;

}

Font-family permite ingresar múltiples opciones cuando el navegador las recorre de izquierda a derecha hasta que encuentra una fuente en el sistema que puede ser usada. Esto significa que siempre debes asegurarte de tener una opción generic-family como tu último valor, ya que si alguno de los otros puede ser encontrado, el navegador utilizará la opción predeterminada para la familia especificada.

* El navegador busca la fuente Helvetica Neue en el sistema de usuario y si la encuentra la utiliza.
* Si no encuentra Helvetica Neue, el navegador busca Arial en el sistema del usuario y si la encuentra la utiliza.
* Si Arial no es encontrada, usará Verdana.
* Como último recurso, si no encuentra ninguna de las fuentes en el grupo de fuentes, volvemos a sans-serif instruyendo al navegador que use sea cual sea la fuente sans-serif predeterminada del sistema. No sabrás que se usará exactamente en este caso, pero al menos es mejor que terminar con la fuente predeterminada del navegador, Times new roman, la cual es una fuente serif.

**Ten en cuenta que las fuentes con más de una palabra en su nombre deben ir entre comillas.**

### font-size:

La propiedad font-size (tamaño de fuente) puede ser expresada en 4 unidades diferentes: pt, px, em, %. Pt y px (punto y pixel) son consideradas dimensionamiento estático y no se ajustarán adecuadamente cuando se redimensione tu página. Em y % (medida responsiva y porcentaje) sí se ajustarán y, por ende,  son las recomendadas por los desarrolladores. A continuación podrás ver un cuadro desplegable que muestra las equivalencias aproximadas entre las cuatro unidades.



h3 {

font-size: 10pt;

}

p {

font-size: 14px;

}

a {

font-size: 1.2em;

}

span {

font-size: 80%;

}

### font-style

Esta propiedad controla la inclinación de la letra. Algunos textos pueden tener la propiedad italic (cursiva) como parte de la fuente, por lo que seleccionarla sería simple. Para las fuentes que no tienen el estilo italic, el valor oblique puede ser utilizado para imitar el texto en italic.

span {

font-style: italic;

}

### font-weight

Esta propiedad define el grosor de una línea de caracteres. normal es normalmente el valor por defecto. Los valores pueden ser establecidos de forma numérica o semántica.



p {

font-weight: bold;

}

span {

font-weight: 600;

}

## ¿Qué es una etiqueta span?

Si quieres resaltar un texto específico dentro de un párrafo o división, entonces puedes utilizar <span>. A diferencia de <div> que sitúa al siguiente <div> en una línea separada, <span> por defecto pone al elemento en la misma línea. Por ejemplo, considera

<h1>Hello <div>World</div> </h1>

Arriba por defecto pondríamos ‘World’ (Mundo) en una nueva línea. Sin embargo, si quisieras que World se mostrara en la misma línea, deberías escribir

<h1>Hello <span>World</span> </h1>

Luego, en el css, podrías hacer algo como esto para resaltar solo la palabra "World":

h1 span {

font-weight: bold;

color: blue;

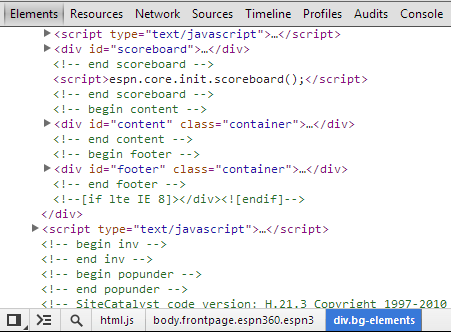
}

Para leer más sobre la diferente entre <div> y <span>, puede leer: <https://www.codecademy.com/en/forum_questions/502ad0ea558dfe0002026d69>

# Cómo usar Inspeccionar Elemento (Inspect Element)

Editar HTML y CSS puede verse como desactivar una bomba, un pequeño cambio y tu proyecto enloquece. Tu mejor aliado al editar son las herramientas de desarrollador (Developer Tools). Se incluyen en **Chrome**, **Firefox**, **la última versión de Internet Explorer (Edge) y Safari.**

Para acceder a Inspeccionar Elemento, simplemente usa el clic derecho en cualquier elemento de la ventana del navegador y cliquea en Inspeccionar (**Inspect**) ¡y listo!. Inspeccionar Elemento es una caracteriza que muestra todas las propiedades del sitio web que estás viendo.



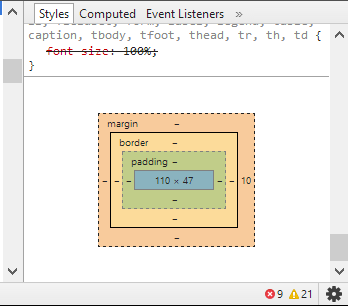
Esta es una captura de pantalla de la ventana inspeccionar de Chrome. En la barra gris de arriba, verás un conjunto de opciones incluyendo ‘elements’ y ‘console’, que muestran distinta información de la página.

## Pestaña de Elementos (Elements Tab)

Por las siguientes dos lecciones, nos centraremos en ESPN.com. Puesto que la mayoría de las empresas cambian sus sitios webs con frecuencia, ten en cuenta que los sitios de ESPN que verás, pueden ser un tanto diferentes a los ejemplos del vídeo de más abajo. Pese a ello, las mismas herramientas y conceptos pueden ser aplicados.

La **pestaña elements**, te permite ver el HTML que está siendo procesado por tu navegador. Puedes ver todas las*classes*e *ids*del contenido HTML, lo que es de gran ayuda para depurar CSS. La imagen de más arriba, muestra la pestaña de elementos de una porción del HTML de ESPN.com ¡Fíjate que incluso puedes ver comentarios!!

Si cliqueas en un elemento particular del HTML dentro del cuadro de **elementos**, verás que el lado derecho de la ventana de Inspeccionar Elemento cambia. Esta parte de la ventana muestra una tonelada de información de **CSS y JavaScript**. Hay una pestaña a la derecha llamada ‘**estilos**’ (styles) que es por lejos la herramienta más útil al trabajar con CSS.



 (Este es el elemento que detallan las 2 imágenes anteriores)

La pestaña **estilos** muestra **todo el el CSS procesado para un elemento particular**. La captura de pantalla anterior es la información de CSS para el logo de ESPN en su página web. A la izquierda de la imagen se muestra los estilos ejecutados del elemento y la hoja de estilo de donde fueron tomados. Baja hasta el final de la pestaña y verás la imagen a la derecha, que debería ser familiar. Esta imagen representa los valores del modelo de cajas del elemento en cuestión. Puedes ver el **margen (margin), relleno (padding), borde (border) y las dimensiones de cualquier elemento de la página**. Aún mejor, puedes pasar por una parte específica de la caja en la ventana de inspeccionar elemento y la propiedad particular del elemento será resaltada en la página ¡Inténtalo! Esta herramienta es genial para ajustar la posición (los cambios solo se realizarán en el navegador y no se guardarán en el archivo CSS)

Mira la imagen nuevamente, verás un sector donde dice ‘**element.style**’. Ahí puedes escribir tu propio CSS y modificar lo que se muestra en la página web. Por supuesto, estos cambios no son permanentes, pero ya puedes imaginarte lo útil de esta herramienta. Otra cosa más que puedes hacer con todas las reglas de estilos de la pestaña estilos (styles), es desactivar las propiedades CSS asociadas des-marcándolas.

## Cómo usar Inspeccionar Elemento

## Genial, ¿y ahora qué?

Cualquier buen desarrollador web te dirá que **Inspeccionar Elemento** es una excelente herramienta en la lucha contra el código deficiente. Hay personas cuyo trabajo es específicamente hacer estas herramientas de desarrollo, por lo que estás en buenas manos. Inspeccionar es muy fácil de entender cuando la exploras ¡Empieza a usarla regularmente y te ahorrará mucho tiempo!

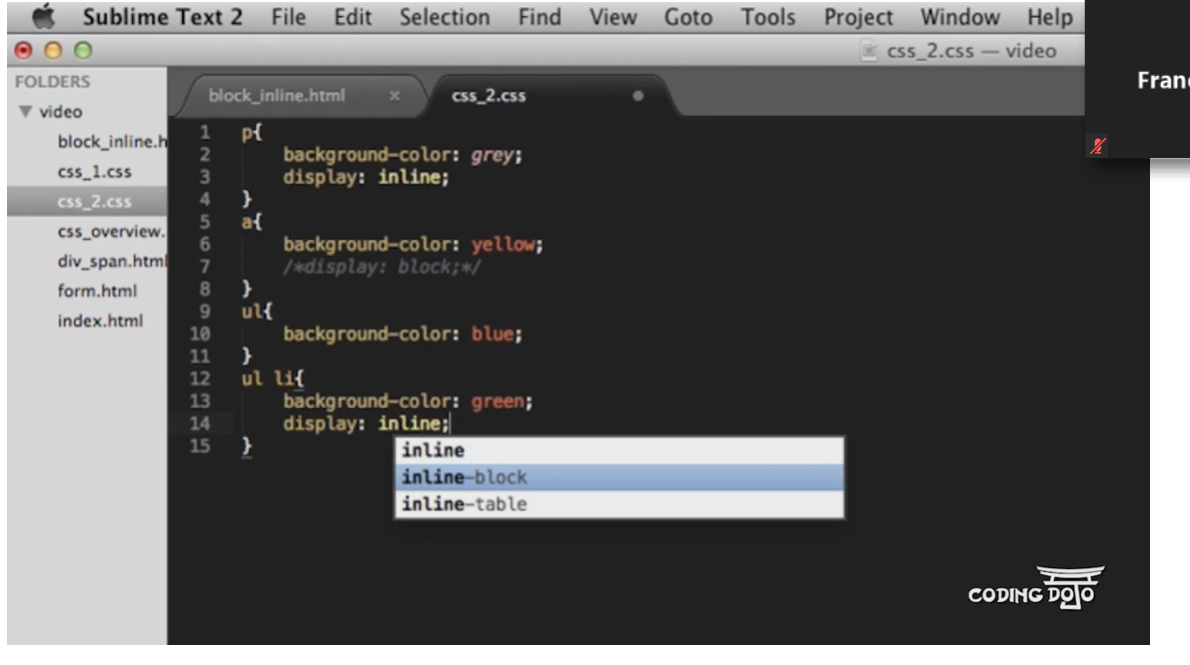
## Selector de Color

El diseño y selección de la paleta de colores representan una parte importante en la creación de una página web. Para ayudar con una selección de colores precisa, hay una variedad de herramientas disponibles para encontrar y definir los colores, además de armar paletas de colores. Es fácil encontrar en internet herramientas para generar esquemas de colores, pero encontrar el color exacto utilizado en el borde del botón de tu sitio web favorito puede ser más complicado. En esta sección esbozaremos cómo usar la herramienta de selector de color (colorpicker) de Chrome Dev, que nos ayudará a encontrar colores que nos interesan desde otros sitios web.

Para  acceder al selector de color, primero necesitamos abrir el Inspector de elementos y seleccionar un elemento con un color de fondo. Luego, cliqueamos en el cuadrado de color de la ventana de Estilo (Styles). Entonces, se mostrará información respecto al color de fondo del elemento, permitiendo distintos ajustes como los valores del código de colores, distintas opciones de colores y una gradiente de intensidad y oscuridad. Desde ahí, es posible encontrar el valor de los colores en la página moviendo el cursor del mouse por la página. Por medio de un círculo se mostrarán los pixeles ampliados. El píxel central del círculo es el que toma el valor del color al hacer clic.



# La Propiedad Display- Block, Inline-Block, Inline



\*Puedes bajar una copia del código demostrado en el vídeo anterior desde el siguiente enlace [aquí (block-inline.zip)](http://s3.amazonaws.com/General_V88/boomyeah/company_209/chapter_2135/handouts/chapter2135_3328_block-inline.zip).

Cada elemento HTML es por defecto inline o block.

Los elementos Inline solo toman cuanto espacio sea necesario.

Elementos Inline: <a>, <img>, <span>, <input>, <label>, <select>, <textarea>.

Los ítems **Block** toman el 100% del ancho del elemento padre (parent). Incluso si el ancho del elemento fuera menor que el padre, de todas formas abarcaría el 100% del ancho.

Elementos Block: <body>, <div>, <form>, <h1>-<h6>, <p>, <table>, <ul>, <ol>, <li>

**No es válido poner elementos block dentro de elementos inline.**

Ej:

No válido:

<a href="http://www.google.com">http://www.google.com</a>">www.google.com<h1>Google</h1></a>

Válido:

<h1><a href="<a href="http://www.google.com">http://www.google.com</a>">Google</a></h1>

Sin embargo, podemos cambiar el estilo de visualización predeterminado de HTML utilizando CSS y la propiedad de visualización (display). Los posibles valores de esta propiedad son: **inline, block, none e inline-block**.

Ej: Enlaces Inline predeterminados

<a href="#">One</a>

<a href="#">Two</a>

<a href="#">Three</a>

<a href="#">Four</a>

<a href="#">Five</a>

a {

margin: 10px;

background: #eee;

color: black;

padding: 10px;

}

Convirtiéndolos en elemento de bloque (block):

<a href="#">One</a>

<a href="#">Two</a>

<a href="#">Three</a>

<a href="#">Four</a>

<a href="#">Five</a>

a {

margin: 10px;

background: #eee;

color: black;

display: block;

padding: 10px;

}

En el reverso, supón que tenemos el siguiente menú de navegación:

<ul id="nav-menu">

<li><a href="#">Home</a></li>

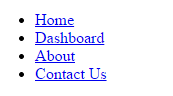
<li><a href="#">Dashboard</a></li>

<li><a href="#">About</a></li>

<li><a href="#">Contact Us</a></li>

</ul>

El HTML normalmente aparecería así:



Cuando queremos hacer un menú horizontal en la parte superior de nuestra página, podemos hacer lo siguiente:

#nav-menu{

list-style-type: none; /\*This gets rid of the bullets.\*/

}

#nav-menu li {

display: inline;

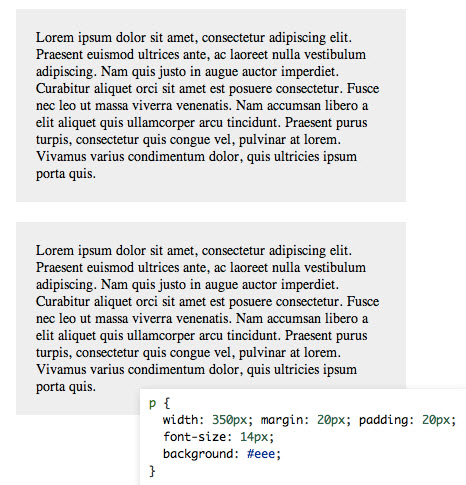
}

Ahora se ve así:



**Inline-block:**

Considera el siguiente ejemplo:



Recuerda que las etiquetas <p> son elementos block por defecto. Cuando las visualizamos como inline, ambos actúan juntos en un solo párrafo.

<p>Lorem ipsum dolor sit amet...</p>

<p>Lorem ipsum dolor sit amet...</p><br>

p {

font-size: 14px;

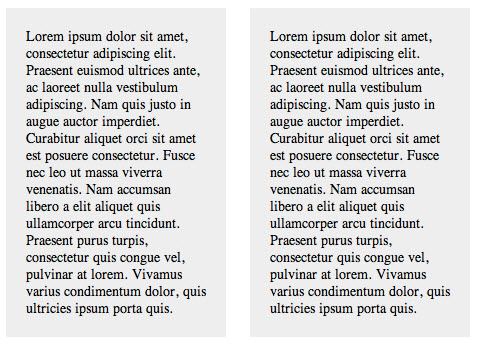
background: #eee;

display: inline;

}

Hemos perdido toda capacidad de establecer nuestro ancha y alto puesto que los dos párrafos están completamente integrados.

Los mismos párrafos con inline-block:



p {

width: 180px;

margin:10px;

padding:20px;

font-size: 14px;

background: #eee;

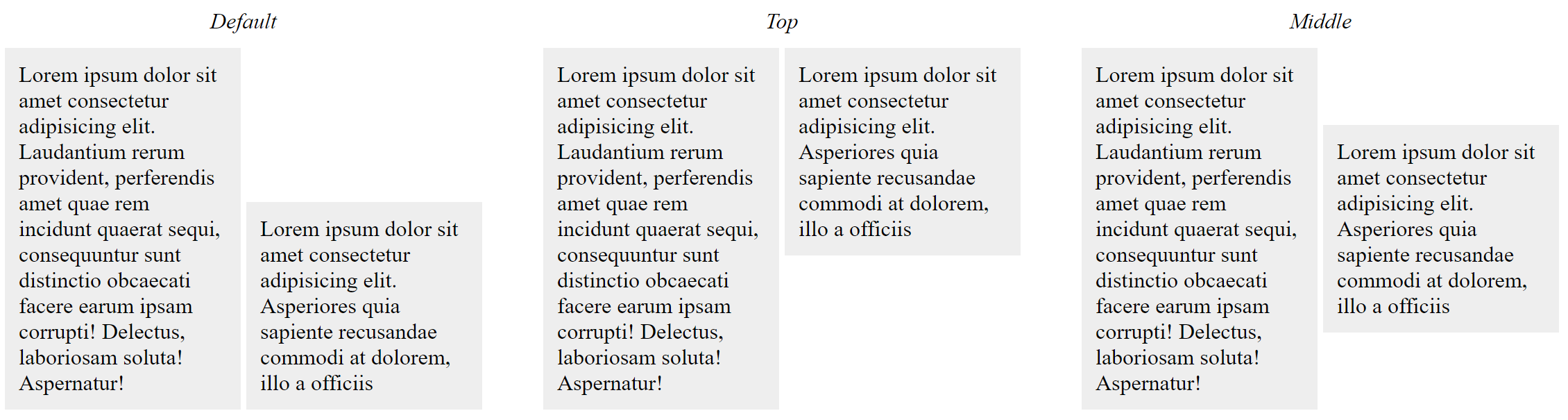
display: inline-block;

}

Lo que sucede aquí es que le estamos pidiendo al navegador que visualice los párrafos inline, pero permitiéndoles mantener sus características block-level. Esto significa que podemos fijar un ancho y alto manualmente y que ambos elementos permanezcan diferentes, además de aparecer uno junto al otro en el flujo del documento.

## Alineación Vertical

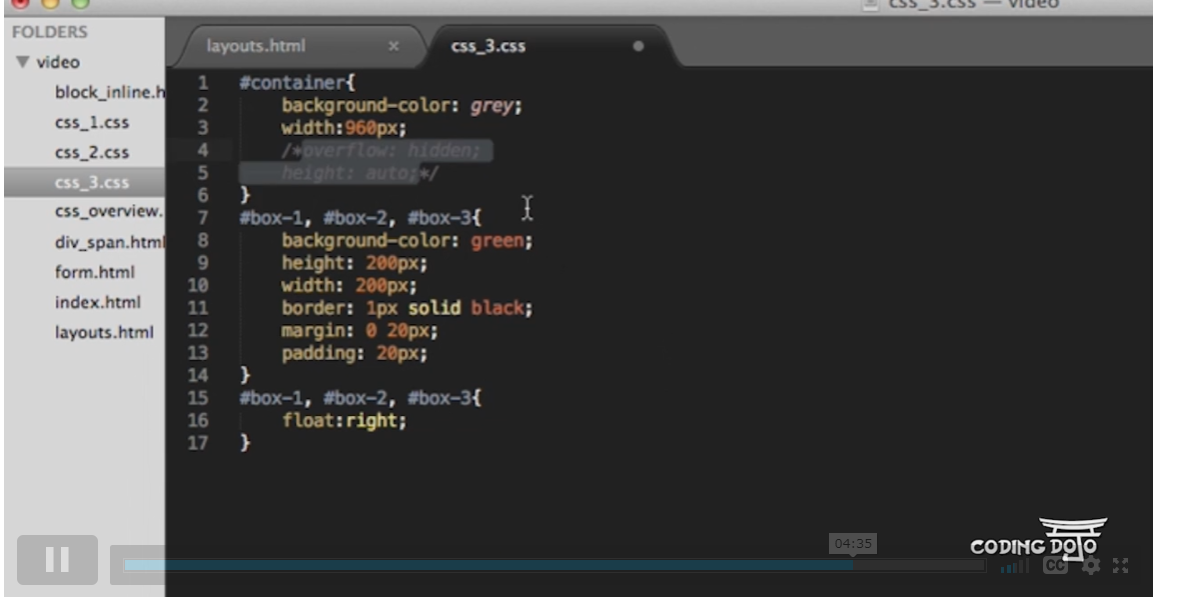
Cuando comenzamos a utilizar inline-block, tal vez habremos notados que la posición vertical de los bloques puede empezar a comportarse de manera diferente a lo que esperábamos, especialmente cuando los bloques tienen diferentes alturas. Por defecto, los elementos se alinean con el borde de más abajo de los bloques dentro de la línea. Para ajustar este comportamiento, podemos usar la propiedad de CSS denominada vertical-align la cual tiene valores como top (arriba) y middle (al medio).



**Lectura Complementaria:**

<https://uniwebsidad.com/libros/css/capitulo-5?from=librosweb>

# Modelo de Caja (Box Model) - Margen (Margin), Borde (Border), Relleno (Padding)



\*Puedes descargar un copia del código anterior [acá (layouts.zip)](http://s3.amazonaws.com/General_V88/boomyeah/company_209/chapter_2135/handouts/chapter2135_3329_layouts.zip).

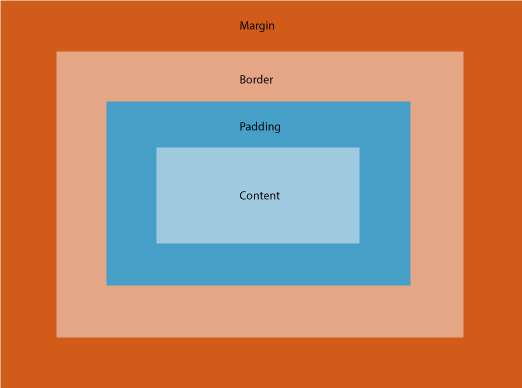
Todos los diseños web se realizan con elementos de bloque o block. Los diseñadores utilizan blocks (normalmente elementos <div>), para crear áreas rectangulares donde caben todos los contenidos. Solo existen 3 reglas:

* **Área total**: El espacio que un elemento ocupa y afecta.
* **Float, clear y overflow**
* **Elementos anidados (nested)**

**Área Total**

El ancho total se refiere a cuánto espacio horizontal utiliza un bloque. Esto incluye el margen, el borde y el relleno del bloque. Calcular el ancho, relleno y margen es a menudo el mayor dolor de cabeza para los diseñadores, pero es fácil ver cómo trabajan si utilizas el **modelo de caja**.

El modelo de caja se compone de las propiedades de **margen (margin), borde (border) y relleno (padding)**.



El margen está afuera de los elementos de bloque, mientras que el relleno está dentro de ellos. Esto significa que usamos el margen para separar un bloque de las que lo rodean, y relleno para mover el contenido de un bloque lejos de sus bordes.

Podemos definir específicamente el margen, relleno o borde de cualquier lado de un elemento.

div {

padding-top: 10px;

padding-right: 10px;

padding-bottom: 10px;

padding-left: 10px;

}

Además puedes usar la propiedad de taquigrafía (shorthand):

div {

padding-top: 25px;

padding-right: 50px;

padding-bottom: 75px;

padding-left: 100px;

}

que es equivalente a:

div {

padding: 25px 50px 75px 100px;

}

Y

div {

padding-top: 25px;

padding-right: 50px;

padding-bottom: 75px;

padding-left: 50px;

}

que es equivalente a:

div {

padding: 25px 50px 75px;

}

Y

div {

padding-top: 25px;

padding-right: 50px;

padding-bottom: 25px;

padding-left: 50px;

}

que es equivalente a:

div {

padding: 25px 50px;

}

Y

div {

padding-top: 25px;

padding-right: 25px;

padding-bottom: 25px;

padding-left: 25px;

}

que es equivalente a:

div{

padding: 25px;

}

(El orden va en el sentido del reloj, arriba -> derecha -> abajo -> izquierda).

Entonces, y de acuerdo al modelo de caja, el ancho total de un elemento es**:**

**(Margin x 2) + (Border x 2) + (Padding x 2) + Content Width (Ancho del contenido)**

Calcular la altura (height) es más complicado ¿Por qué? Porque los márgenes **verticales se desploman**.

Ej:

HTML:

<div id="box-1">

</div>

<div id="box-2">

</div>

<div id="box-3">

</div>

CSS:

#box-1, #box-2, #box-3{

height: 100px;

width: 100px;

background-color: red;

}

#box-1{

margin: 20px;

}

#box-2{

margin: 30px;

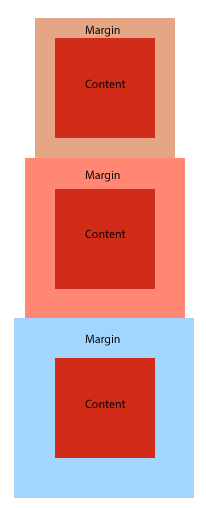
}

#box-3{

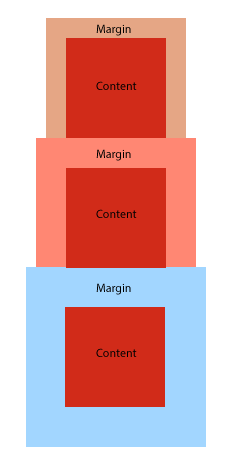
margin: 40px;

}

Tal vez piensas que esto se verá así:



Pero el código se verá más bien así:



Cuando los márgenes verticales de dos elementos se tocan, **solo el margen del elemento con el valor de margen más grande se mantendrá**, mientras que el margen del elemento con el menor valor de margen se reducirá a cero.

Existen otras situaciones donde los elementos no tienen tienen un derrumbe o reducción en sus márgenes:

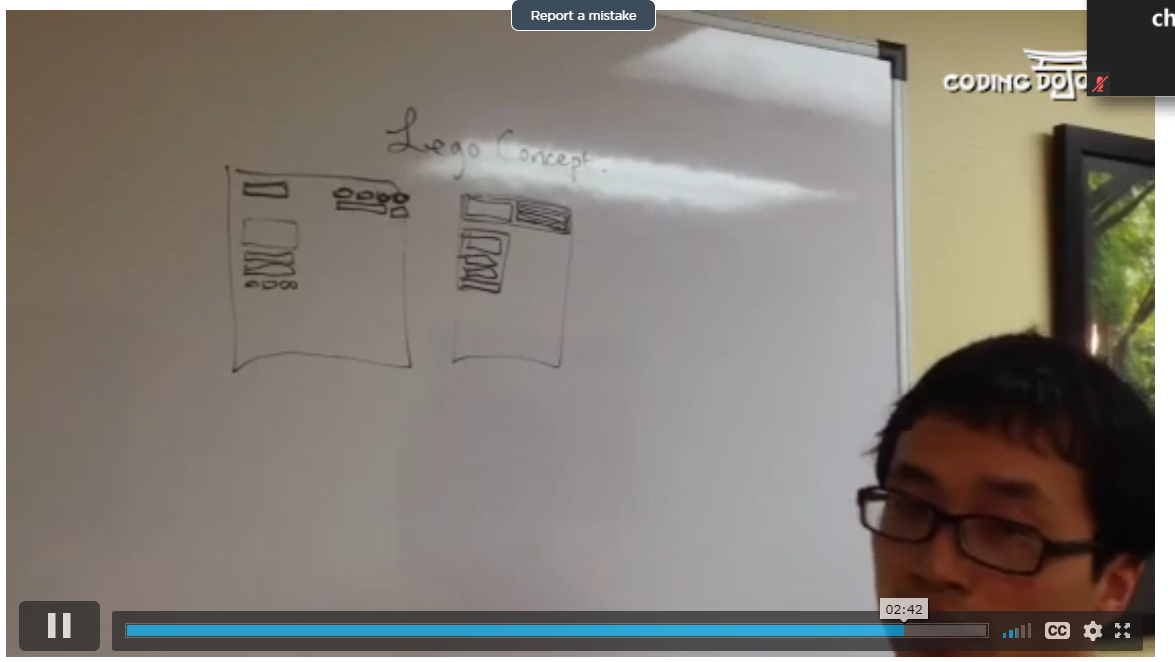
* Elementos flotantes
* Elementos absolutamente posicionados
* Elementos inline-block
* Elementos con la propiedad overflow establecida para cualquier otra cosa, menos para que sea visible (Estos no derrumban sus márgenes con sus hijos).
* Elementos eliminados o cleared (Estos no derrumban sus márgenes superiores con el margen inferior de su bloque padre).

#### **Quiz**

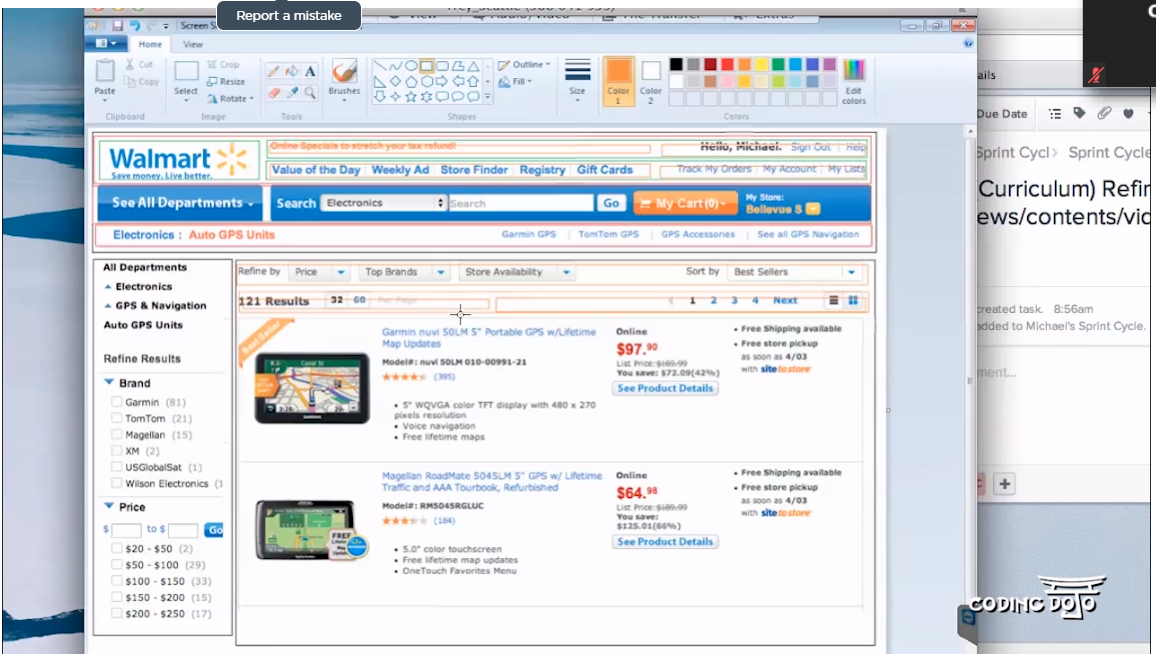
1. ¿Qué etiquetas de HTML deberías usar para conectar tu archivo HTML con su correspondientes archivo CSS externo?
   *  <meta name="css" src="style.css">
   *  <script type="text/css" src="style.css">
   *  <link rel="stylesheet" type="text/css" href="style.css">
   *   Ninguna de las Anteriores
2. ¿Cuál de estas sintaxis no funcionará?
   *  display: none
   *  display: static
   *  display: inline
   *  display: block
3. ¿Para qué es útil la herramienta de desarrollador del navegador?
   *  Monitorear las solicitudes y respuestas de HTTP desde una página web.
   *  Ver y editar as propiedades de HTML y CSS
   *  Depurar Javascript y AJAX
   *  Todas las anteriores
4. ¿Qué método de incrustación (embedding) de CSS se recomienda?
   *  internal embedding (añadir una etiqueta style en el head de html)
   *  external embedding (añadir una etiqueta link tag en el head de html)
   *  inline embedding (añadir style='…' en la etiqueta html tag)
   *   Todas las anteriores
5. Verdadero o Falso: El elemento “input” puede ser también un selector CSS
   *  Verdadero. Puedes agregar estilos a un elemento input y a un tipo específico de input.
   *  Falso. Puedes no agregar ningún estilo CSS a un elemento input.
   *  Verdadero. Pero solo podrás poner estilos según tipo de input (ej: input[type=submit] { } ).

# Concepto Lego

El concepto Lego es una forma de visualizar los diferentes elementos en tu sitio como piezas de lego. Así, primero colocarás los grandes bloques y luego bloques más pequeños dentro de ellos. Al focalizarse en los bloques grandes primero, puedes pensar en el diseño general de tu sitio. Luego de posicionar los bloques grandes, debes determinar qué bloques más pequeños existirán dentro de los bloques más grandes y situarlos de acuerdo a ello. Puede parecer más rápido mirar una captura de pantalla y empezar a escribir en HTML/CSS inmediatamente, pero dibujar las diferentes piezas de lego en la pizarra primero (las grandes, luego las más pequeñas) te ayudará a construir en HTML/CSS de manera más expedita.

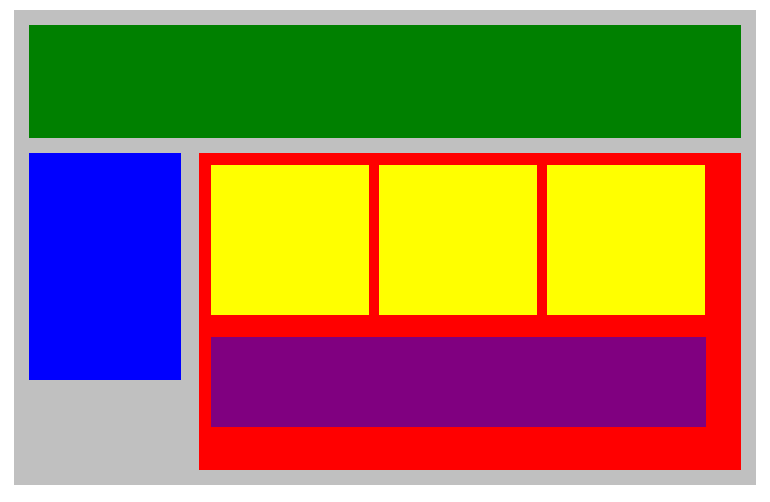


## Aplicando el Concepto Lego a un Sitio (WalMart)



**Trazando tus Bloques**

Intenta duplicar la siguiente imagen ajustando el código CSS. Usa **margins y paddings** para ajustar los espacios entre divisiones y utiliza la propiedad display para lograr situar cada bloque en el lugar correcto. Tal vez necesites propiedades de CSS adicionales.



Este es el código de HTML:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Position Practice</title>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<div id="wrapper">

<div id="header"></div>

<div id="navigation"></div>

<div id="main\_content">

<div class="subcontents"></div>

<div class="subcontents"></div>

<div class="subcontents"></div>

<div id="advertisement"></div>

</div>

</div><!-- end of wrapper -->

</body>

</html>

y CSS:

/\*CSS reset settings here\*/

\*{

margin: 0px;

padding: 0px;

}

#wrapper{

width: 950px;

background-color: silver;

margin: 0px auto;

}

#header{

min-height: 150px;

background-color: green;

}

#navigation{

min-height: 300px;

width: 200px;

background-color: blue;

}

#main\_content{

min-height: 400px;

width: 700px;

background-color: red;

}

.subcontents{

min-height: 200px;

width: 210px;

background-color: yellow;

}

#advertisement{

min-height: 120px;

width: 660px;

background-color: purple;

}

*No gastes más de 2-3 horas en esta actividad.*

Mientras realizas esta actividad, utiliza min-height (altura mínima) y vertical align (alineación vertical) para darle una altura mínima a la separación y alinear verticalmente algunos de los inline-blocks. Por favor, NO uses float (utiliza display:inline-block).

# La Propiedad de Posición (Position Property)

Principio del formulario

Final del formulario

Además de la propiedad de visualización (display property) y el modelo de caja, la propiedad position también puede ser utilizada para mover elementos en una página. Utiliza esta propiedad solo cuando lo necesites, de lo contrario, usa la propiedad display y el modelo de caja para posicionar tus elementos. La especificación de CSS nos ofrece 6 valores de posición, sin embargo, cubriremos las 4 más importantes: **static** (estático), **relative** (relativo), **absolute** (**absoluto** y **fixed** (fijo)). Cada propiedad sirve a un propósito específico, por lo que entenderlo es la clave para dominar los diseños basados en CSS.

## ****Considera lo siguiente:****

HTML:

<div id="container">

<div id="box-1">

</div>

<div id="box-2">

</div>

<div id="box-3">

</div>

</div>

CSS:

#container{

background-color: yellow;

display: inline-block;

}

#box-1, #box-2, #box-3{

height: 100px;

width: 100px;

margin: 10px;

display: inline-block;

}

#box-1{

background-color: green;

}

#box-2{

background-color: pink;

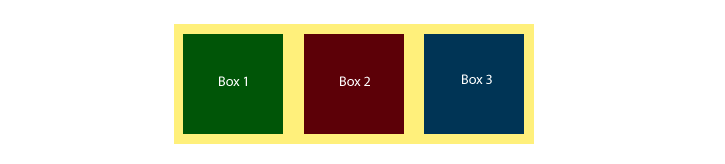
}

#box-3{

background-color: blue;

}

Esto resulta en:



## ****Static****

El posicionamiento estático (static) viene predeterminado. Para cualquier elemento que tenga posición, static es parte del flujo de documentos normal. Las reglas sobre dónde se ubica y cómo afecta otras cajas o cuadros está definido por el modelo de caja (puedes revisar nuevamente la lección sobre el **modelo de caja**).

Un elemento situado de forma estática no tomará en cuenta ningún valor para las propiedades top (arriba), right (derecha), bottom (abajo) y left (izquierda), como tampoco ninguna declaración z-index. Para poder usar tales propiedades, tu elemento que necesariamente usar posicionamiento absolute, relative o fixed.

## ****Absolute****

Los elementos posicionados de forma absoluta (absolute) están completamente eliminados del flujo normal de documentos. En cuanto a los elementos que lo rodean, el elemento posicionado absolutamente no existe. Es como si la propiedad de visualización (display) del elemento estuviera fijada en “ninguna”. Entonces, si quieres mantener el espacio para que otros elementos no lo utilicen, debes seguir otros caminos.

Para fijar la ubicación de un elemento absolutamente posicionado se utilizan las propiedades top, right, bottom y left. Normalmente definirás solo 2, ya sea top o bottom, o left o right. Por defecto, cada una tiene un valor de ajuste automático.

La clave para entender el posicionamiento absoluto, es comprender cuál es el origen. Si top es fijado en 20px, la pregunta sería: ¿20px de dónde?

**Un elemento posicionado de manera absoluta (absolute) se ubica en relación al primer elemento padre (parent), el cual tiene una posición distinta de la estática aplicada a este. Si ningún elemento padre cumple esta condición, el elemento de posicionamiento absoluto se ubica en relación a la ventana del documento.**

Utilizando el posicionamiento absolute en el box-1:

#container{

background-color: yellow;

display: inline-block;

}

#box-1, #box-2, #box-3{

height: 100px;

width: 100px;

margin: 10px;

display: inline-block;

}

#box-1{

background-color: green;

position: absolute;

top: 40px;

}

#box-2{

background-color: pink;

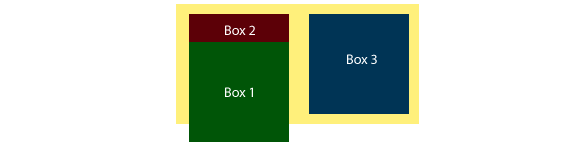
}

#box-3{

background-color: blue;

}

Da como resultado:



## ****Relative****

Los elementos posicionados de forma relativa (relative) se ubican de acuerdo a las mismas propiedades top, right, bottom y left, las cuales pueden cambiarse fácilmente. En cierto sentido, agregar posicionamiento relative es similar a agregar un margen, pero con una importante diferencia: los elementos alrededor de un elemento posicionado de manera relativa actúan como si ese cambio no existiera. Lo ignoran.

Usando posicionamiento relative en el box-2:

#container{

background-color: yellow;

display: inline-block;

}

#box-1, #box-2, #box-3{

height: 100px;

width: 100px;

margin: 10px;

display: inline-block;

}

#box-1{

background-color: green;

}

#box-2{

background-color: pink;

position: relative;

top: 20px;

left: 20px;

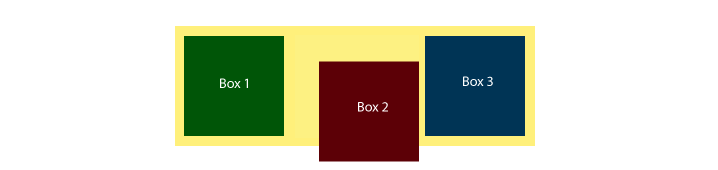
}

#box-3{

background-color: blue;

}

Da como resultado:



## ****Fixed****

El posicionamiento fixed o fijo, actúa de manera similar al posicionamiento absoluto con un par de diferencias. Primero, el elemento de posicionamiento fixed siempre se posiciona en función de la ventana del navegador y toma los ya ahora conocidas propiedades top, right, bottom y left. La segunda diferencia es inherente al nombre, los elementos posicionadas de manera fija (fixed) están fijos. No se mueven a medida que se desplaza la página.

## ****Extras****

Un recurso genial para saber cómo centrar elementos: <http://designshack.net/articles/css/how-to-center-anything-with-css/>

Una forma fácil de centrar un elemento dentro de un contenedor (container):

HTML:

<div id="container">

<div id="center-me">

</div>

</div>

CSS:

#container{

width:200px;

height:200px;

background: blue;

position:relative;

}

#center-me{

background: red;

height: 100px;

width: 100px;

margin: auto;

position: absolute;

left: 0;

bottom: 0;

top: 0;

right: 0;

}

Otra forma de centrar horizontalmente un elemento:

HTML:

<div id="container">

<div id="center-me">

</div>

</div>

CSS:

#container{

width:200px;

height:200px;

background: blue;

position:relative;

}

#center-me{

background: red;

height: 100px;

width: 100px;

margin: 0 auto;

}

# Compatibilidad del Navegador

Cada navegador puede entregar atributos predeterminados a los elementos HTML, provocando que los sitios se vean diferentes dependiendo de la versión del navegador o si estás viendo el sitio en IE, Firefox, Chrome, etc. Por ejemplo, en Internet Explorer, un elemento H1 puede tener cierto margen (margin) y relleno (padding) haciendo que tu sitio se vea de una forma, mientras que Chrome y Firefox pueden darle a H1 un margen y relleno ligeramente diferente. Estas diferencias podrían ser un dolor de cabeza, especialmente si estás tratando de hacer que tu sitio se vea consistente en los distintos navegadores o versiones de navegador.

Para resolver los problemas entre navegadores, es recomendable utilizar ‘RESET’ y ‘NORMALIZE’ y definir cuáles debieran ser los atributos predeterminados (en vez de confiar en los valores predeterminados que vienen con cada navegador). De hecho, a los reclutadores les encanta preguntar sobre esto y ver si sabes como hacer que tu sitio se vea consistente en los diferentes navegadores. Cuando te pregunten esto, deberías poder explicar (con seguridad) qué significa reset/normalizing y por qué se utilizan. Además, deberías contarles que es importante validar  tu HTML/CSS a menudo, puesto que hacer esto te da una idea de cómo arreglar inconvenientes que podrían ocasionar problemas de compatibilidad entre navegadores (y frecuentemente, un HTML/CSS inválido puede conducir a un comportamiento extraño puesto que el navegador no arreglaría el HTML/CSS inválido de forma correcta).

## CSS Reset

Estas es una muy buena explicación de los CSS resets, por qué los usamos y algunos ejemplos:

<http://perishablepress.com/a-killer-collection-of-global-css-reset-styles/>

## Normalize

Una alternativa a los CSS resets, ampliamente utilizada, es normalize.css, que también cubre los elementos de HTML5. Para más información respecto a lo que hace y por qué puede ser una mejor opción que los CSS resets, ingresa aquí:

<https://github.com/necolas/normalize.css/>

# Preguntas Frecuentes

**P: ¿Puedo agregar múltiples clases a un elemento?**

**R: Sí.** Simplemente separa los nombres de clase con un espacio. Ej:

<div class="nav side-bar"></div>

**P: ¿Puedo agregar una clase y un id al mismo elemento?**

**R: Sí.** Un elemento puede tener un id y múltiples clases. Ej:

<div id="primary-nav" class="nav side-bar"></div>

**P: ¿Tengo que agregar una clase o un id para apuntar a un elemento anidado?**

**R: No como una regla**. Considera el siguiente ejemplo:

<ul id="nav-menu">

<li><a href="#">Home</a></li>

<li><a href="#">Dashboard</a></li>

<li><a href="#">About</a></li>

<li><a href="#">Contact Us</a></li>

</ul>

Para focalizarse en los enlaces (links) y eliminar el subrayado de estos, podemos hacer lo siguiente:

#nav-menu li a{

text-decoration: none;

}

Esto le dice al navegador que primero ubique los elementos con id nav-menu, luego que encuentre toda la lista de ítems dentro de las listas desordenadas y que, finalmente, encuentre los enlaces dentro de dichas listas.

**Por lo general, debes hacer tu código más fácil de leer evitando agregar demasiadas clases e ids.**