EXERCISES QUE TRABAJAREMOS EN EL CUE

- EXERCISE 1: MODELO DE CAJAS.
- EXERCISE 2: TIPOS DE CAJAS, DISPLAY Y VISIBILITY.
- EXERCISE 3: ELEMENTOS FLOTANTES.

EXERCISE 1: MODELO DE CAJAS.

En el siguiente ejercicio aprenderás a utilizar el modelo de cajas o en inglés, box model.

MODELO DE CAJAS O BOX MODEL

En CSS, cada elemento HTML se encuentra envuelto en una caja, la cual determina el diseño y despliegue de éstos en una página web. Dicha caja cuenta con cuatro componentes: los márgenes (`margin`), los bordes (`border`), el relleno (`padding`) y el contenido en sí; éstos determinan el tamaño y cómo se ubicará en el espacio un elemento HTML.

- *Contenido:* se refiere al contenido de la caja, donde aparece el texto y las imágenes.
- *Padding o relleno:* corresponde a un contorno transparente que rodea al contenido.
- *Bordes:* corresponde a un borde que se encuentra entre el `padding` y los márgenes. A este se le puede asignar un color.
- *Márgenes:* al igual que el `padding`, es un contorno transparente, que se ubica por fuera de los bordes.

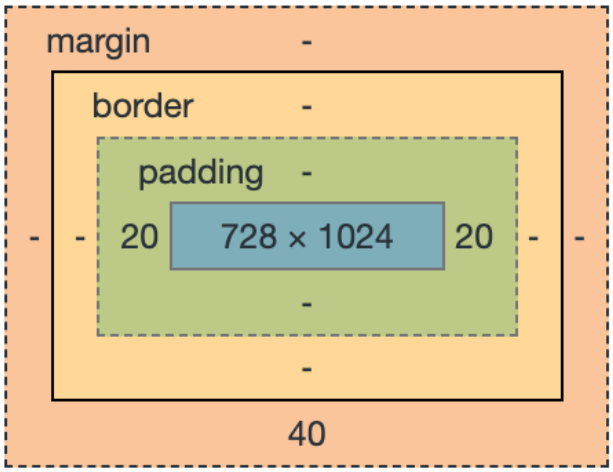

Imagen 1. Box model o modelo de cajas.

Revisemos un ejemplo, y para esto vamos a utilizar el proyecto que creamos anteriormente en el segundo ejercicio del CUE 2 de Sass.

Crearemos una etiqueta `<div>` sin ningún contenido debajo de la etiqueta `<h1>`, y le agregaremos el atributo `class="box"`. A continuación, levantaremos el proyecto con live server, nos dirigiremos al inspector de elementos del navegador y los revisaremos.

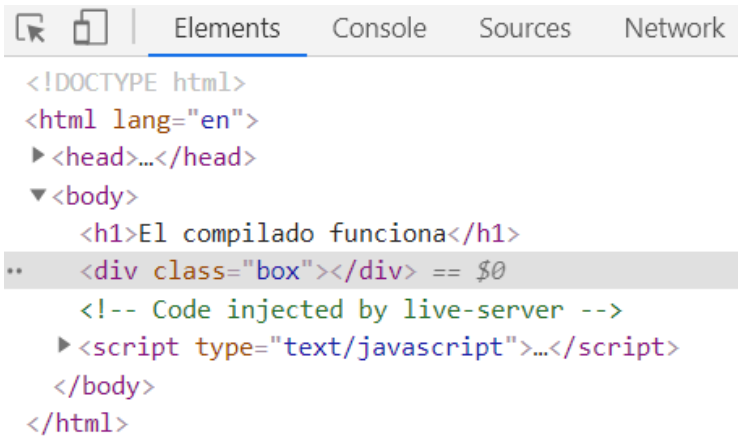

Imagen 2. Ejemplo div.

Ya con el div creado en el documento HTML, le agregaremos algunos estilos.

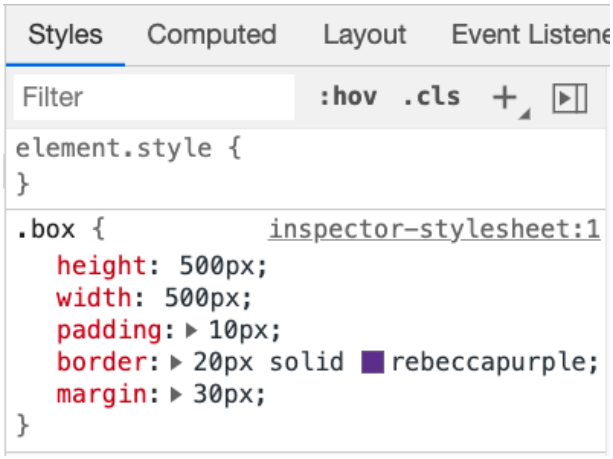

Imagen 3. Hoja de estilos.

Finalmente, revisaremos el modelo de cajas de este div y su tamaño, el cual corresponderá a `content-box`; es decir, el tamaño (`height` y `width`) de la caja corresponderá al tamaño del contenido de ésta (500px * 500px en este caso).

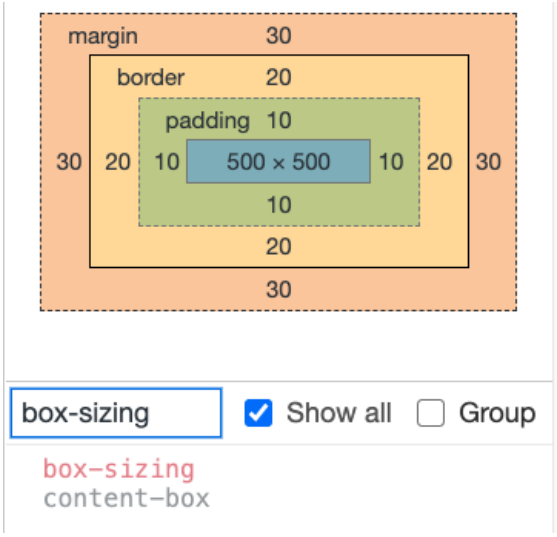

Imagen 4. Modelo de cajas.

CSS3 permite cambiar las propiedades de un elemento HTML por defecto, por lo que podemos modificar el valor de `box-sizing` a `border-box`. Ésto le indicará al navegador que los valores de `margin`, `border`, `padding` y `content` deben sumar `500px` de alto y ancho.

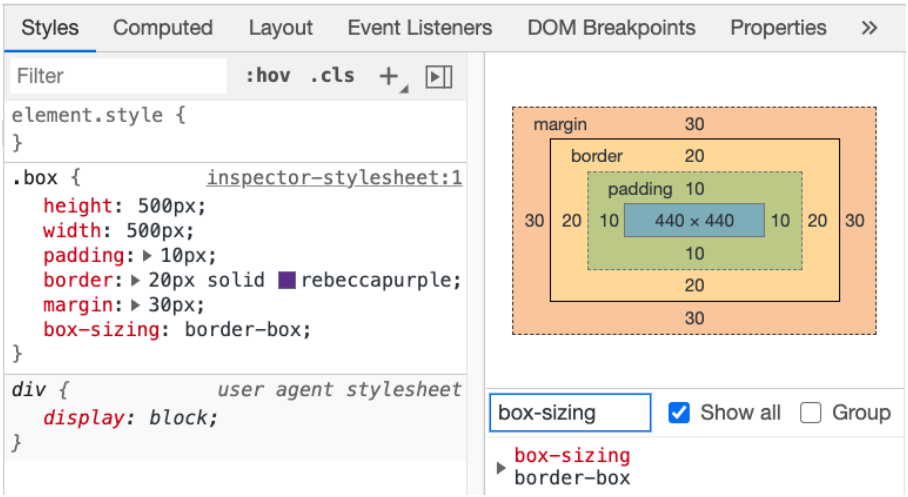

Imagen 5. Ejemplo propiedad `box-sizing: border-box;`

Importante: si la propiedad `box-sizing` no es soportada por el navegador, debemos utilizar los `vendor prefixes` (`-moz`, `-webkit`, `-ms`, `-o`). Estos prefijos son palabras que se colocan antes de una propiedad de CSS para que ésta pueda ser interpretada correctamente. En este caso, quedarían así:

- `-moz-box-sizing: border-box;` (Firefox).
- `-webkit-box-sizing: border-box;` (Android, Chrome, iOS y Safari).
- `-ms-box-sizing: border-box;` (Edge e Internet Explorer).
- `-o-box-sizing: border-box;` (Opera y Opera Mini).

Existen herramientas, como Autoprefixer y Can I Use, que facilitan la creación de prefijos.

En este ejercicio hemos aprendido sobre el Modelo de Cajas. Te invitamos a seguir realizando las actividades correspondientes a este CUE y a continuar avanzando en el curso.

EXERCISE 2: TIPOS DE CAJAS, DISPLAY Y VISIBILITY.

En el siguiente ejercicio aprenderemos acerca de los tipos de cajas (elementos en bloque y en línea), de igual manera, sobre propiedades como display y visability.

TIPOS DE CAJAS: PROPIEDAD DISPLAY

Es la más importante en CSS para controlar la disposición de los elementos. Existen tres valores de la propiedad `display: inline`, `block` e `inline-block`. La mayoría de los elementos tienen como valor por defecto `block` o `inline`.

Utilizaremos el siguiente ejemplo para entender mejor esta propiedad:

```
<!DOCTYPE html>
<html lang="es">
▶ <head>…</head>
▼ <body>
    <div class="box">Soy una caja</div>
    <div class="box">Soy otra caja</div> == $0
  </body>
</html>
```

Imagen 6. Ejemplo propiedad display.

ELEMENTOS INLINE

Éstos comienzan en la misma línea que el elemento anterior a ellos, y su ancho corresponde al mínimo posible para poder mostrar su contenido. Algunos ejemplos son: `<span>`, `<a>`, `<em>` y `<strong>`.

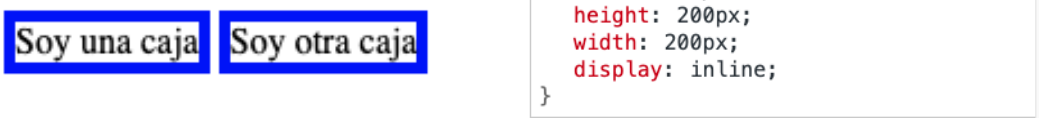

Imagen 7. Elementos con `display: inline;`.

Como se puede observar en la Imagen 7, al utilizar la propiedad `inline`, los elementos se posicionan uno al lado del otro (se distribuyen de manera horizontal), y no se respeta el alto y ancho de `200px` que se definieron como estilo, sino que solo se utilizó el tamaño mínimo necesario para desplegar su contenido.

ELEMENTOS BLOCK

Éstos comienzan siempre en una nueva línea y ocupan todo el ancho disponible, es decir, se extienden horizontalmente lo máximo posible. Algunos ejemplos son: `<div>`, `<h1>-<h6>`, `<p>`, `<form>`, `<header>`, `<footer>` y `<section>`.

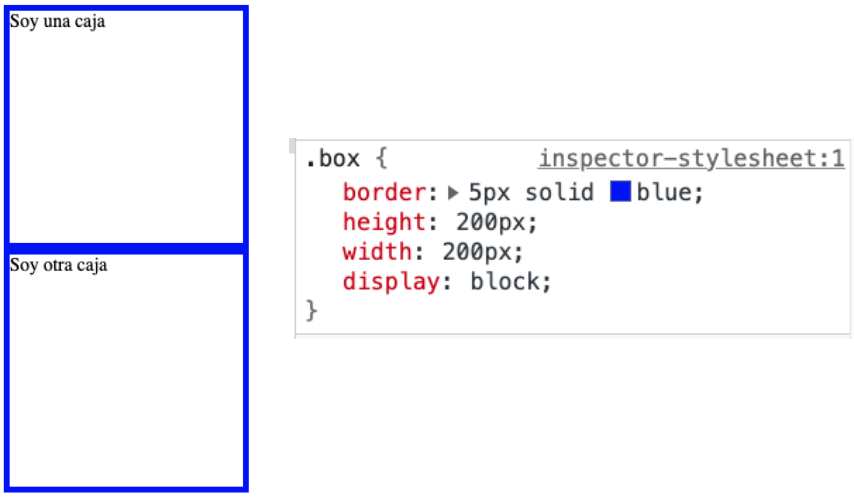

Imagen 8. Elementos con `display: block;`.

En este caso, los elementos se posicionan uno debajo del otro, respetando el ancho y alto definido en la hoja de estilos.

ELEMENTOS INLINE-BLOCK

Éstos se refieren a un híbrido entre `block` e `inline`. A diferencia de los elementos block, estos no agregan un salto de línea después del elemento, es decir, pueden ubicarse junto a otros. Además, a diferencia de los elementos `inline`, estos permiten establecer un ancho y una altura del elemento. También respetan los márgenes y `paddings`. Un ejemplo son las imágenes, que se despliegan una al lado de la otra y se les puede definir un ancho y alto.

Imagen 9. Elementos con `display: inline-block;`.

La propiedad `display: inline-block;`, permite que los elementos se alineen horizontalmente, pero, a diferencia de `display: inline`, se respeta el ancho y alto definido.

DISPLAY: NONE;

Esta propiedad hace “desaparecer” un elemento de nuestra maqueta, pero éste solo es un efecto visual, ya que, si inspeccionamos nuestra página web, el elemento seguirá ahí.

Imagen 10. Elementos con `display: none;`.

Existe otra propiedad muy útil para esconder elementos:

VISIBILITY

Ésta permite esconder elementos que no se quieren mostrar al usuario, pero manteniendo su espacio dentro de la maqueta, es decir, deja un espacio vacío donde debería estar.

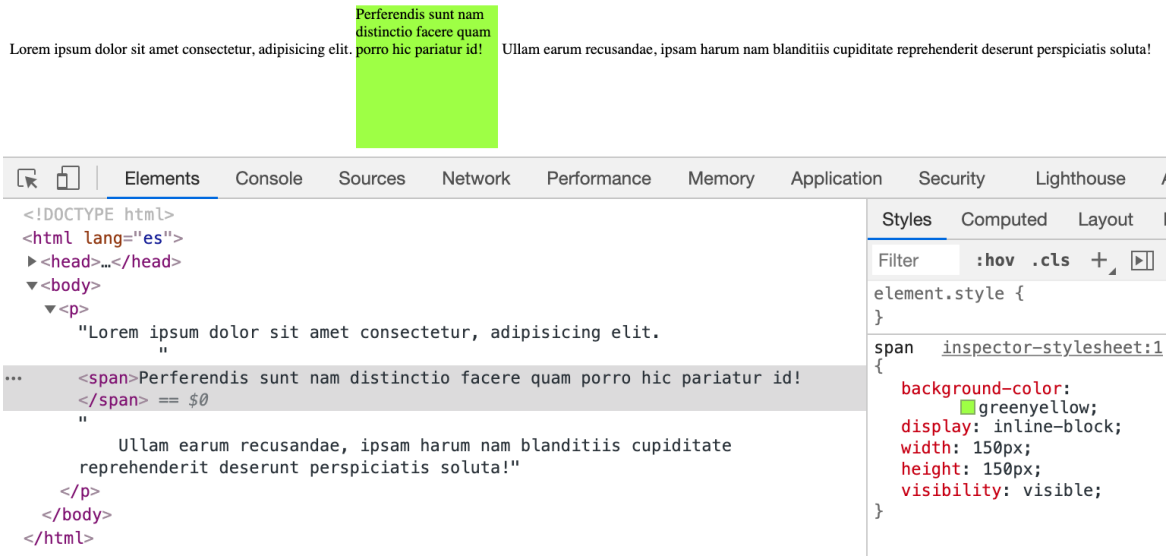

Imagen 11. Ejemplo `visibility: visible;`.

En el ejemplo de la Imagen 11, cambiaremos el valor de la propiedad `visibility` por `hidden`, lo cual esconderá todo el contenido del elemento `span`.

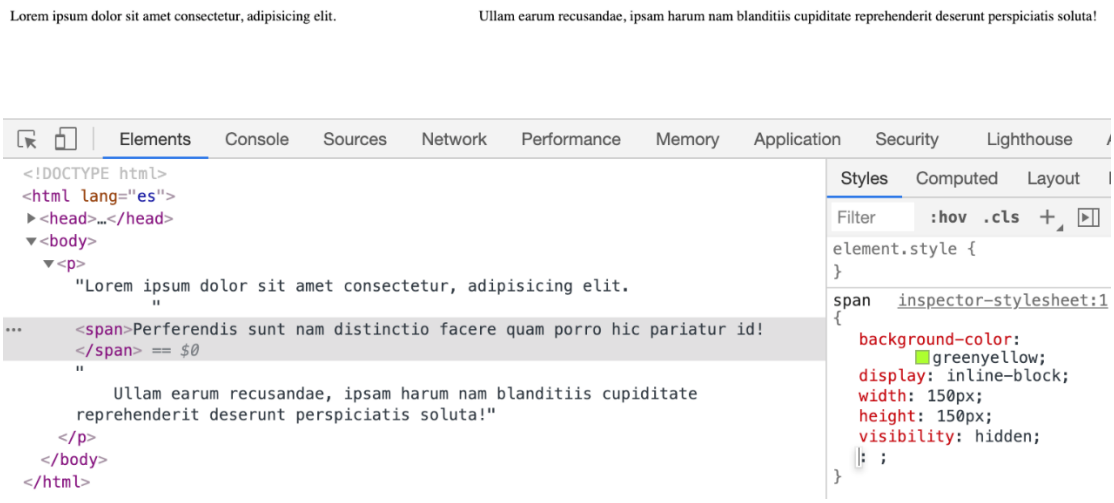

Imagen 12. Ejemplo `visibility: hidden;`.

En este ejercicio hemos aprendido sobre los tipos de cajas que existen en CSS y las propiedades `display` y `visibility`. Te invitamos a seguir realizando las actividades correspondientes a este CUE y a continuar avanzando en el curso.

EXERCISE 3: ELEMENTOS FLOTANTES.

En el siguiente ejercicio aprenderás a utilizar las propiedades `float` y `clear`.

FLOAT

Esta propiedad permite sacar elementos del flujo normal de HTML y llevarlos a la izquierda o derecha, haciendo que el resto del elemento se posicione al lado de éste, lo que permite ordenar de mejor manera nuestro contenido.

Crearemos una página web sencilla, con una imagen y un texto.

```html
<!DOCTYPE html>
<html lang="es">
   <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-
scale=1.0">
      <title>Box model</title>
      <link rel="stylesheet" href="assets/css/style.css">
   </head>
   <body>
      <div class="container">
         <img src="assets/img/24969178668_b87343d74d_o.jpg"
            width=”180px” alt="Elefantes">
         <p>Lorem ipsum dolor sit amet consectetur adipisicing
elit. Consequuntur consectetur nam eum alias in sunt
            tempora, vitae esse, temporibus libero repellendus.
Dolor sequi, deserunt provident beatae voluptates
            repellat qui accusantium.
         </p>
      </div>
   </body>
</html>
```

```css
.container {
       background-color: gold;
}
```

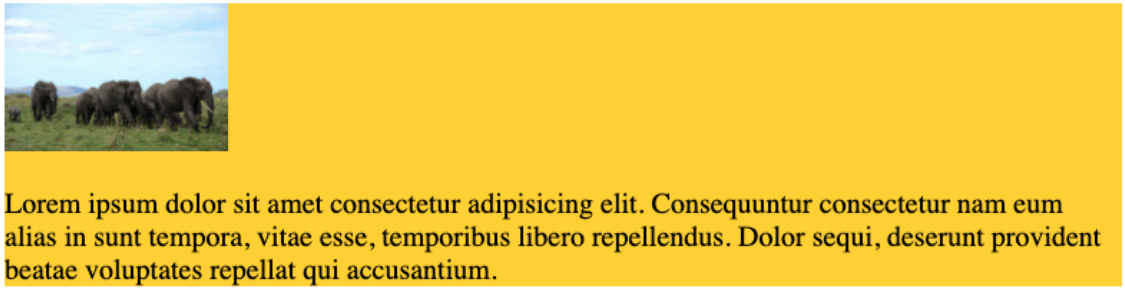

Imagen 1. Ejemplo al cual se le aplicará la propiedad `float`.

Por defecto, los elementos `img` y `p` se despliegan en bloque, uno abajo del otro. Con `float right` haremos que la imagen quede a la misma altura del texto.

```
<!DOCTYPE html>
<html lang="es">
   <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width,
initial-scale=1.0">
      <title>Box model</title>
      <link rel="stylesheet" href="assets/css/style.css">
   </head>
   <body>
      <div class="container">
         <img class="float-right"
src="assets/img/24969178668_b87343d74d_o.jpg" width="180px"
alt="Elefantes">
         <p>Lorem ipsum dolor sit amet consectetur adipisicing
elit. Consequuntur consectetur nam eum alias in sunt
            tempora, vitae esse, temporibus libero repellendus.
Dolor sequi, deserunt provident beatae voluptates
            repellat qui accusantium.
         </p>
      </div>
   </body>
</html>
```

```css
.container {
        background-color: gold;
}

.float-right {
        float: right;
}
```

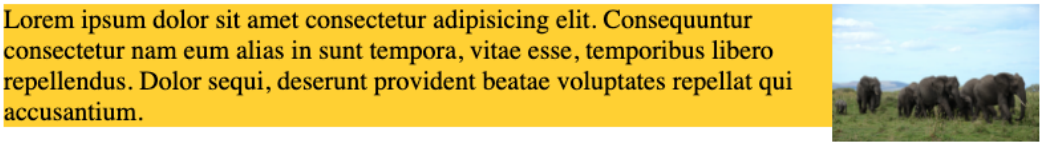

Imagen 2. Ejemplo con propiedad `float: right;`.

Al utilizar la propiedad `float`, la imagen sale del flujo normal de HTML, posicionándose lo más a la derecha posible. Pero hay un detalle: el color de fondo solo cubre el tamaño del texto, sin importar las dimensiones de la imagen. Esto sucede porque al salir la imagen del flujo normal, para el padre ésta ya no existe, por lo que el ancho y alto depende exclusivamente del texto.

CLEAR

Ésta permite limpiar un elemento afectado por `float`. Se pueden utilizar los valores `left`, `right` o `both`.

Para entenderla, agrandaremos un poco la imagen anterior y agregaremos un segundo párrafo.

Imagen 3. Ejemplo sin propiedad `clear`.

Como podrás observar, la propiedad `float` de la imagen afecta a ambos párrafos. Agregaremos `clear: right` al segundo bloque de texto, obteniendo que este párrafo vuelve a su posición original.

```html
<!DOCTYPE html>
<html lang="es">
   <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-
scale=1.0">
      <title>Box model</title>
      <link rel="stylesheet" href="assets/css/style.css">
   </head>
   <body>
      <div class="container">
         <img class="float-right"
src="assets/img/24969178668_b87343d74d_o.jpg"    width="180px"
alt="Elefantes">
         <p>Lorem ipsum dolor sit amet consectetur adipisicing
elit. Consequuntur consectetur nam eum alias in sunt
            tempora, vitae esse, temporibus libero repellendus.
Dolor sequi, deserunt provident beatae voluptates
            repellat qui accusantium.
         </p>
         <p class="clear-right">Lorem ipsum dolor sit amet
consectetur adipisicing elit. Fuga quisquam impedit doloribus
nesciunt, commodi blanditiis ullam eos asperiores eaque vitae
beatae perferendis dolorum iste quasi, corrupti sit sed?
Expedita, cumque!</p>
      </div>
   </body>
</html>
```

```css
.container {
       background-color: gold;
}

.float-right {
       float: right;
}

.clear-right {
       clear: right;
}
```

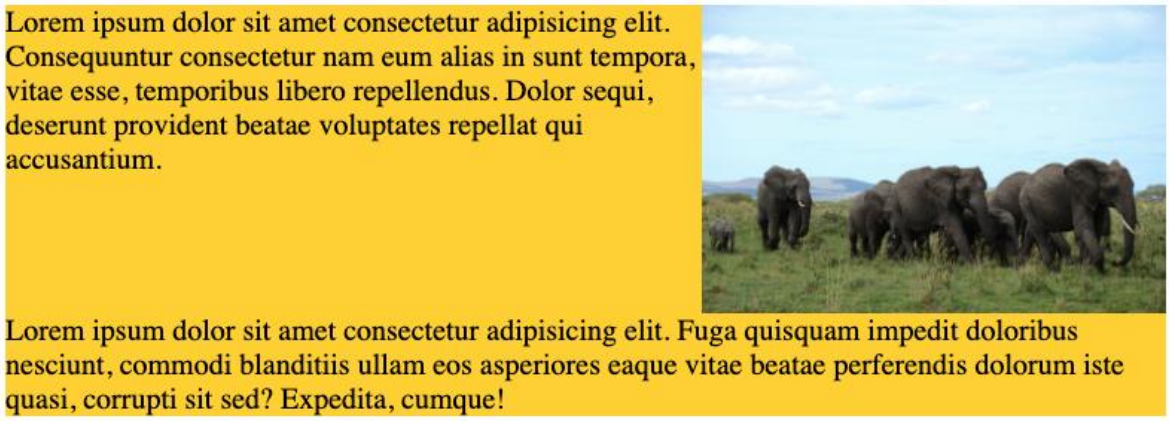

Imagen 4. Ejemplo con propiedad `clear: right;`.

En este ejercicio hemos aprendido sobre los elementos flotantes. Te invitamos a seguir realizando las actividades correspondientes a este CUE y a continuar avanzando en el curso.