24 de octubre de 2022

Coordenadas

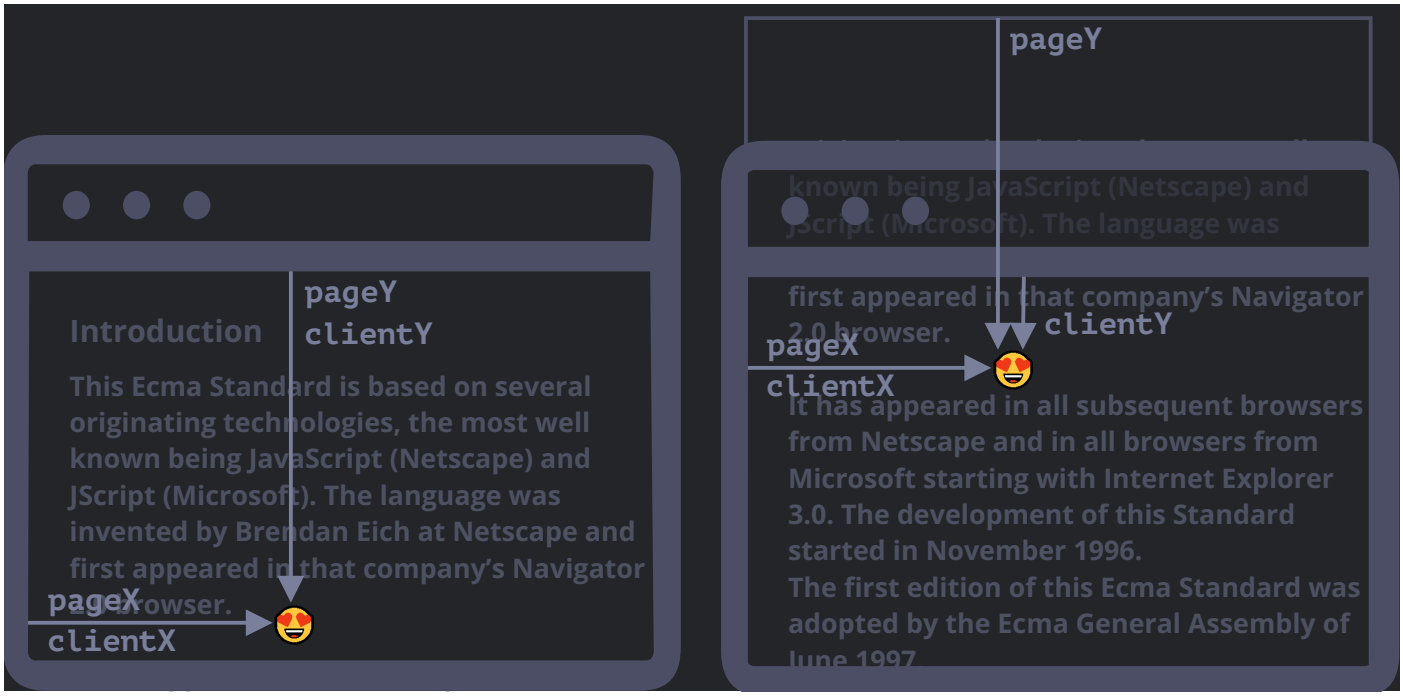
Para mover elementos debemos estar familiarizados con las coordenadas.

La mayoría de los métodos de JavaScript tratan con uno de dos sistemas de coordenadas:

1. **Relativo a la ventana**: similar a `position:fixed`, calculado desde el borde superior/izquierdo de la ventana.
   - Designaremos estas coordenadas como `clientX/clientY`, el razonamiento para tal nombre se aclarará más adelante, cuando estudiemos las propiedades de los eventos.
2. **Relative al documento** – similar a `position:absolute` en la raíz del documento, calculado a partir del borde superior/izquierdo del documento.
   - Las designaremos como `pageX/pageY`.

Cuando la página se desplaza hasta el comienzo, de modo que la esquina superior/izquierda de la ventana es exactamente la esquina superior/izquierda del documento, estas coordenadas son iguales entre sí. Pero después de que el documento cambia, las coordenadas relativas a la ventana de los elementos cambian, a medida que los elementos se mueven a través de la ventana, mientras que las coordenadas relativas al documento permanecen iguales.

En esta imagen tomamos un punto en el documento y demostramos sus coordenadas antes del desplazamiento (primera imagen) y después (segunda imagen):



Cuando el documento se desplazó:

La coordenada `pageY` relativa al documento se mantuvo igual, se cuenta desde la parte superior del documento (ahora desplazada).

- La coordenada `clientY` relativa a la ventana cambió (la flecha se acortó), ya que el mismo punto se acercó a la parte superior de la ventana.

Coordenadas de elemento: getBoundingClientRect

El método `elem.getBoundingClientRect()` devuelve las coordenadas de la ventana para un rectángulo mínimo que encasilla a `elem` como un objeto de la clase interna DOMRect.

Propiedades principales de `DOMRect` :

- `x/y` : coordenadas X/Y del origen del rectángulo con relación a la ventana.
- `width/height` : ancho/alto del rectángulo (pueden ser negativos).

Adicionalmente existen estas propiedades derivadas:

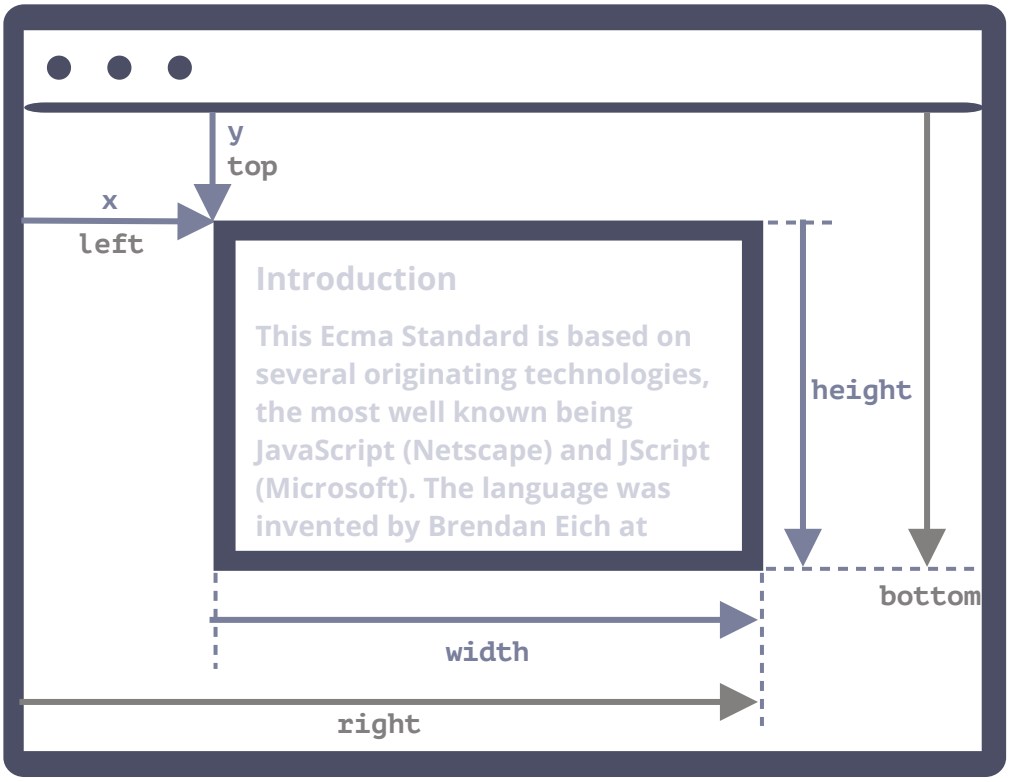
- `top/bottom` : coordenada Y para el borde superior/inferior del rectángulo.
- `left/right` : coordenada X para el borde izquierdo/derecho del rectángulo.

Por ejemplo, haz click en este botón para ver las coordenadas en relación a la ventana:

Recibe las coordenadas para este botón con button.getBoundingClientRect()

Si desplazas la página y repites te darás cuenta que así como cambia la posición del botón relativa a la ventada también cambian sus coordenadas en la ventana (`y/top/bottom` si es que haces scroll vertical).

Aquí hay la imagen con el output de `elem.getBoundingClientRect()` :



Como puedes ver `x/y` y `width/height` describen completamente el rectángulo. Las propiedades derivadas pueden ser calculadas a partir de ellas:

- `left = x`
- `top = y`
- `right = x + width`
- `bottom = y + height`

Toma en cuenta:

- Las coordenadas pueden ser fracciones decimales, tales como `10.5`. Esto es normal ya que internamente el navegador usa fracciones en los cálculos. No tenemos que redondearlos para poder asignarlos a `style.left/top`.
- Las coordenadas pueden ser negativas. Por ejemplo, si la página se desplaza hasta que `elem` rebase el borde superior de la ventana, entonces `elem.getBoundingClientRect().top` será negativo.

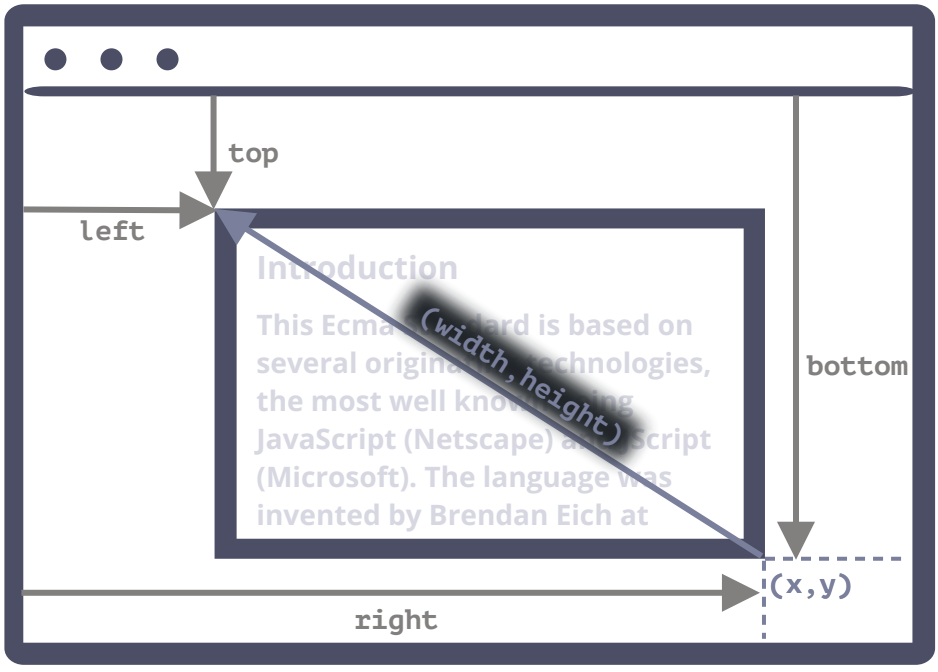
¿Por qué se necesitan propiedades derivadas? ¿Por qué `top/left` si existe `x/y`?

Matemáticamente un rectángulo se define de únicamente con su punto de partida `(x,y)` y el vector de dirección `(width,height)`. Por lo tanto, las propiedades derivadas adicionales son por conveniencia.

Técnicamente es posible que `width/height` sean negativos, lo que permite un rectángulo "dirigido". Por ejemplo, para representar la selección del mouse con su inicio y final debidamente marcados.

Los valores negativos para `width/height` indican que el rectángulo comienza en su esquina inferior derecha y luego se extiende hacia la izquierda y arriba.

Aquí hay un rectángulo con valores `width` y `height` negativos(ejemplo: `width=-200`, `height=-100`):



Como puedes ver: `left/top` no es igual a `x/y` en tal caso.

Pero en la práctica `elem.getBoundingClientRect()` siempre devuelve el ancho y alto positivos. Aquí hemos mencionado los valores negativos para `width/height` solo para que comprendas por qué estas propiedades aparentemente duplicadas en realidad no lo son.

En Internet Explorer no hay soporte para `x/y`

Internet Explorer no tiene soporte para las propiedades `x/y` por razones históricas.

De manera que podemos crear un polyfill y (obtenerlo con `DomRect.prototype`) o solo usar `top/left`, ya que son siempre las mismas que `x/y` para `width/height` positivos, en particular en el resultado de `elem.getBoundingClientRect()`.

Las coordenadas right/bottom son diferentes a las propiedades de posición en CSS

Existen muchas similitudes obvias entre las coordenadas relativas a la ventana y `position:fixed` en CSS.

Pero en el posicionamiento con CSS, la propiedad `right` define la distancia entre el borde derecho y el elemento y la propiedad `bottom` supone la distancia entre el borde inferior y el elemento.

Si echamos un vistazo a la imagen anterior veremos que en JavaScript esto no es así. Todas las coordenadas de la ventana se cuentan a partir de la esquina superior izquierda, incluyendo estas.

elementFromPoint(x, y)

La llamada a `document.elementFromPoint(x, y)` devuelve el elemento más anidado dentro de las coordenadas de la ventana `(x, y)`.

La sintaxis es:

```
let elem = document.elementFromPoint(x, y);
```

Por ejemplo, el siguiente código resalta y muestra la etiqueta del elemento que ahora se encuentra en medio de la ventana:

```
let centerX = document.documentElement.clientWidth / 2;
let centerY = document.documentElement.clientHeight / 2;

let elem = document.elementFromPoint(centerX, centerY);

elem.style.background = "red";
alert(elem.tagName);
```

Debido a que utiliza las coordenadas de la ventana, el elemento puede ser diferente dependiendo de la posición actual del scroll.

Para coordenadas fuera de la ventana, el `elementFromPoint` devuelve `null`

El método `document.elementFromPoint(x,y)` solo funciona si `(x,y)` se encuentra dentro del área visible.

Si alguna de las coordenadas es negativa o excede el ancho o alto de la ventana entonces devolverá `null`.

Aquí hay un error típico que podría ocurrir si no nos aseguramos de ello:

```
let elem = document.elementFromPoint(x, y);
// si las coordenadas sobrepasan la ventana entonces elem = null
elem.style.background = ''; // ¡Error!
```

Usándolas para posicionamiento “fijo”

La mayoría del tiempo necesitamos coordenadas para posicionar algo.

Para mostrar algo cercano a un elemento podemos usar `getBoundingClientRect` para obtener sus coordenadas y entonces CSS `position` junto con `left/top` (o `right/bottom`).

Por ejemplo, la función `createMessageUnder(elem, html)` a continuación nos muestra un mensaje debajo de `elem`:

```
let elem = document.getElementById("coords-show-mark");

function createMessageUnder(elem, html) {
  // Crea un elemento de mensaje
  let message = document.createElement('div');
  // Lo mejor es usar una clase css para el estilo aquí
  message.style.cssText = "position:fixed; color: red";

  // Asignando las coordenadas, no olvides "px"!
  let coords = elem.getBoundingClientRect();

  message.style.left = coords.left + "px";
  message.style.top = coords.bottom + "px";

  message.innerHTML = html;

  return message;
}

// Uso:
// agregarlo por 5 segundos en el documento
let message = createMessageUnder(elem, '¡Hola, mundo!');
document.body.append(message);
setTimeout(() => message.remove(), 5000);
```

Pulsa el botón para ejecutarlo:

Botón con el id="coords-show-mark", el mensaje aparecerá aquí debajo

El código puede ser modificado para mostrar el mensaje a la izquierda, derecha, abajo, aplicando animaciones con CSS para "desvanecerlo" y así. Es fácil una vez que tenemos todas las coordenadas y medidas del elemento.

Pero nota un detalle importante: cuando la página se desplaza, el mensaje se aleja del botón.

La razón es obvia: el elemento del mensaje se basa en `position:fixed`, esto lo reubica al mismo lugar en la ventana mientras se desplaza.

Para cambiar esto necesitamos usar las coordenadas basadas en el documento y `position:absolute`.

Coordenadas del documento

Las coordenadas relativas al documento comienzan en la esquina superior izquierda del documento, no de la ventana.

En CSS las coordenadas de la ventana corresponden a `position:fixed` mientras que las del documento son similares a `position:absolute` en la parte superior.

Podemos usar `position:absolute` y `top/left` para colocar algo en un lugar determinado del documento, esto lo reubicará ahí mismo durante un desplazamiento de página. Pero primero necesitamos las coordenadas correctas.

No existe un estándar para obtener las coordenadas de un elemento en un documento. Pero es fácil de codificarlo.

Los dos sistemas de coordenadas están relacionados mediante la siguiente fórmula:

- `pageY` = `clientY` + el alto de la parte vertical desplazada del documento.
- `pageX` = `clientX` + el ancho de la parte horizontal desplazada del documento.

La función `getCoords(elem)` toma las coordenadas de la ventana de `elem.getBoundingClientRect()` y agrega el desplazamiento actual a ellas:

```
// obteniendo las coordenadas en el documento del elemento
function getCoords(elem) {
  let box = elem.getBoundingClientRect();

  return {
    top: box.top + window.pageYOffset,
    right: box.right + window.pageXOffset,
    bottom: box.bottom + window.pageYOffset,
    left: box.left + window.pageXOffset
  };
}
```

Si el ejemplo anterior se usara con `position:absolute` entonces el mensaje podría permanecer cerca del elemento durante el desplazamiento.

La función modificada `createMessageUnder`:

```
function createMessageUnder(elem, html) {
  let message = document.createElement('div');
  message.style.cssText = "position:absolute; color: red";

  let coords = getCoords(elem);

  message.style.left = coords.left + "px";
  message.style.top = coords.bottom + "px";

  message.innerHTML = html;

  return message;
}
```

Resumen

Cualquier punto en la página tiene coordenadas:

1. Relativas a la ventana: `elem.getBoundingClientRect()` .
2. Relativas al documento: `elem.getBoundingClientRect()` mas el desplazamiento actual de la página.

Las coordenadas de la ventana son ideales para usarse con `position:fixed` , y las coordenadas del documento funcionan bien con `position:absolute` .

Ambos sistemas de coordenadas tienen pros y contras; habrá ocasiones en que ocuparemos una u otra, justamente como con los valores `absolute` y `fixed` para `position` en CSS.

Tareas

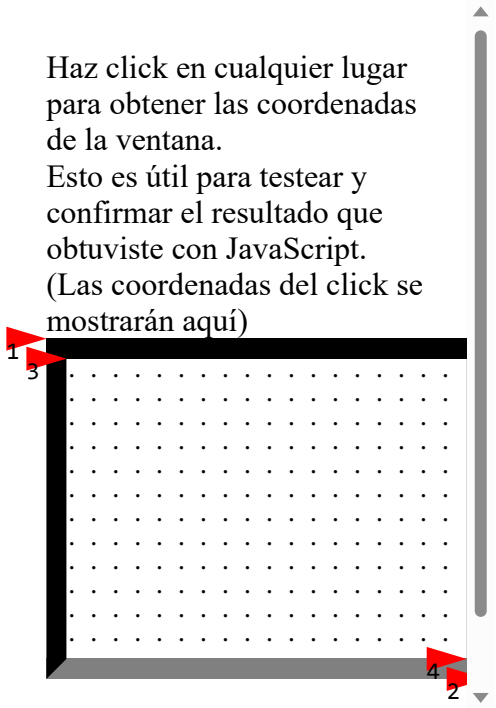
Encuentra las coordenadas del campo en la ventana

importancia: 5

En el siguiente iframe puedes ver un documento con el “campo” verde.

Usa JavaScript para encontrar las coordenadas de las esquinas de la ventana señaladas con las flechas.

Hay una pequeña característica implementada en el documento para conveniencia. Un click en cualquier lugar mostrará las coordenadas ahí.



Tu código debe usar el DOM para obtener las coordenadas en la ventana de:

1. La esquina superior izquierda externa (eso es simple).
2. La esquina inferior derecha externa (simple también).
3. La esquina superior izquierda interna (un poco más difícil).
4. La esquina inferior derecha interna (existen muchas maneras, elige una).

Las coordenadas que tú calcules deben ser iguales a las devueltas por el click del mouse.

P.D. El código también debe funcionar si el elemento tiene otro tamaño o borde, no está ligado a ningún valor fijo.

Abrir un entorno controlado para la tarea.

solución

Muestra una nota cercana al elemento

importancia: 5

Crea una función `positionAt(anchor, position, elem)` que posicione `elem`, dependiendo de la proximidad de `position` al elemento `anchor`.

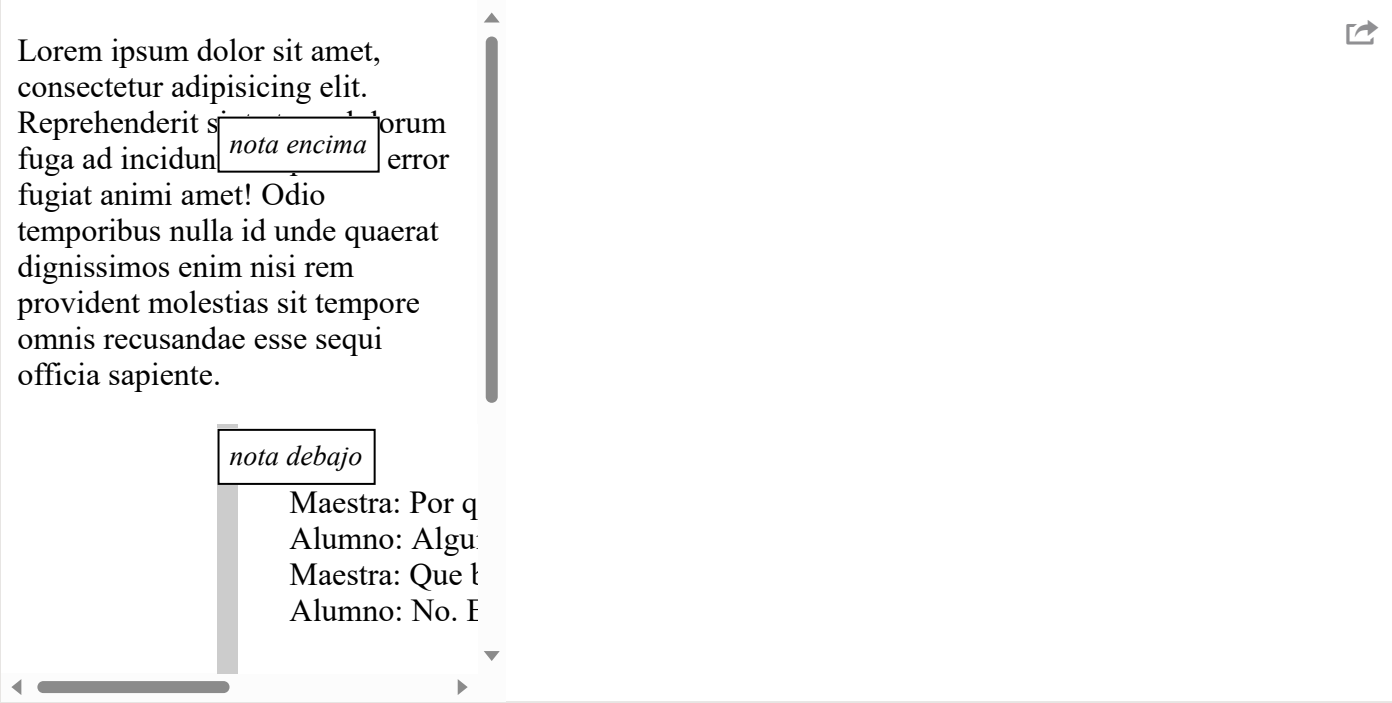
`position` debe ser un string con alguno de estos 3 valores:

- `"top"` – posiciona `elem` encima de `anchor`
- `"right"` – posiciona `elem` inmediatamente a la derecha de `anchor`
- `"bottom"` – posiciona `elem` debajo de `anchor`

Esto será usado dentro de la función `showNote(anchor, position, html)`, proveída en el código fuente de la tarea, que crea un elemento "note" con el `html` y lo muestra en el lugar proporcionado por `position`

cercano a `anchor` .

Aquí está el demo de las notas:



Abrir un entorno controlado para la tarea.

solución

Muestra una nota cercana al elemento (absolute)

importancia: 5

Modifica la solución de la tarea previa de manera que la nota use `position:absolute` en lugar de `position:fixed` .

Esto evitará que se "aleje" del elemento cuando se desplace la página.

Toma la solución de la tarea anterior como punto de partida. Para testear el scroll, agrega el estilo `<body style="height: 2000px">` .

solución

Posiciona la nota adentro (absolute)

importancia: 5

Ampliando a la tarea anterior Muestra una nota cercana al elemento (absolute): enséñale a la función `positionAt(anchor, position, elem)` a insertar `elem` dentro de `anchor` .

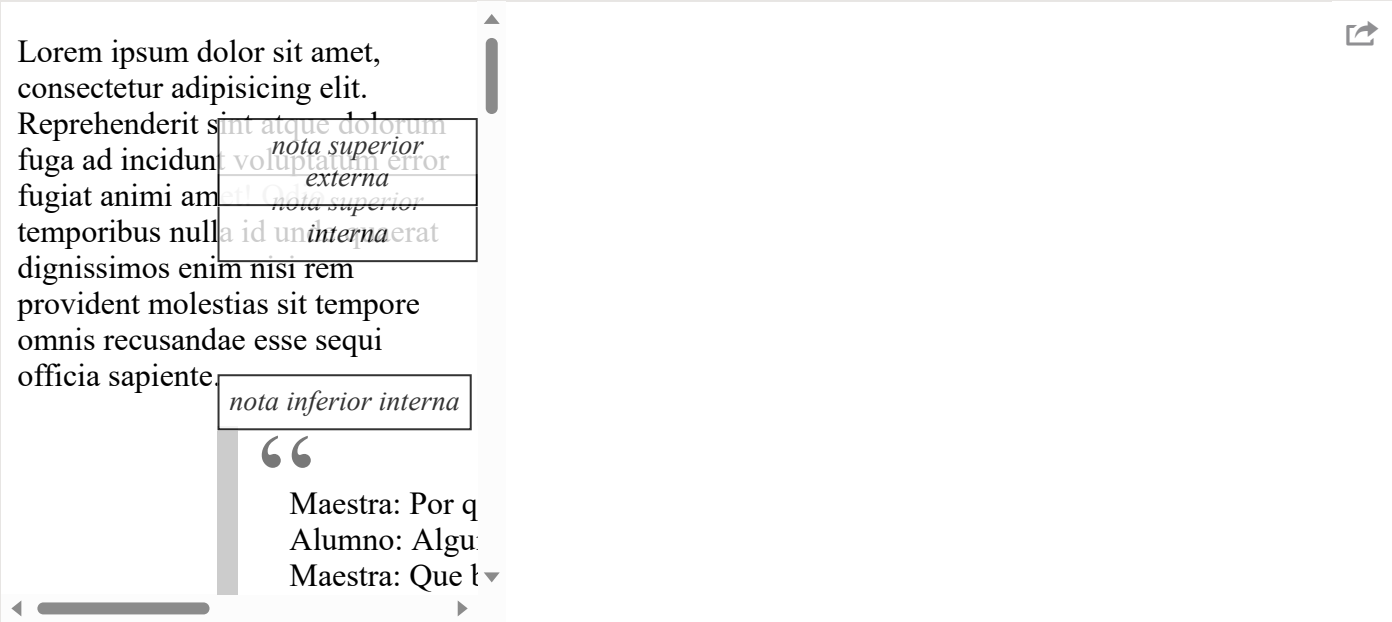
Los nuevos valores para posición son `position` :

- `top-out`, `right-out`, `bottom-out` – funciona ingual que antes, inserta el `elem` encima, a la derecha o debajo de `anchor`.
- `top-in`, `right-in`, `bottom-in` – inserta el `elem` dentro del `anchor`: lo fija en la parte superior, derecha o inferior del borde.

Por ejemplo:

```
1  // Muestra la nota encima de la cita textual
2  positionAt(blockquote, "top-out", note);
3
4  // Muestra la nota dentro de la cita textual en la parte superior
5  positionAt(blockquote, "top-in", note);
```

El resultado:



Para el código fuente toma la solución de la tarea Muestra una nota cercana al elemento (absolute).

solución

Lección anterior

Próxima lección

Compartir

Mapa del Tutorial

Comentarios

- Si tiene sugerencias sobre qué mejorar, por favor enviar una propuesta de GitHub o una solicitud de extracción en lugar de comentar.

- Si no puede entender algo en el artículo, por favor explique.
- Para insertar algunas palabras de código, use la etiqueta `<code>`, para varias líneas – envolverlas en la etiqueta `<pre>`, para más de 10 líneas – utilice una entorno controlado (sandbox) (plnkr, jsbin, codepen…)