UT2 – Lenguajes para visualización de la información

CSS3 – 3 de 3

1. ADVERTENCIA

- Las siguientes transparencias exponen propiedades del lenguaje CSS en su versión 3.
- Las transparencias carecen en su mayoría de ejemplos
  - Se pretende que las pruebes mientras las estudias
- Puedes probarlas mediante un editor de texto y un navegador
  - Notepad++
  - Geany
  - Visual Studio Code

"me lo contaron y lo olvidé, lo vi y lo entendí, lo hice y lo aprendí."

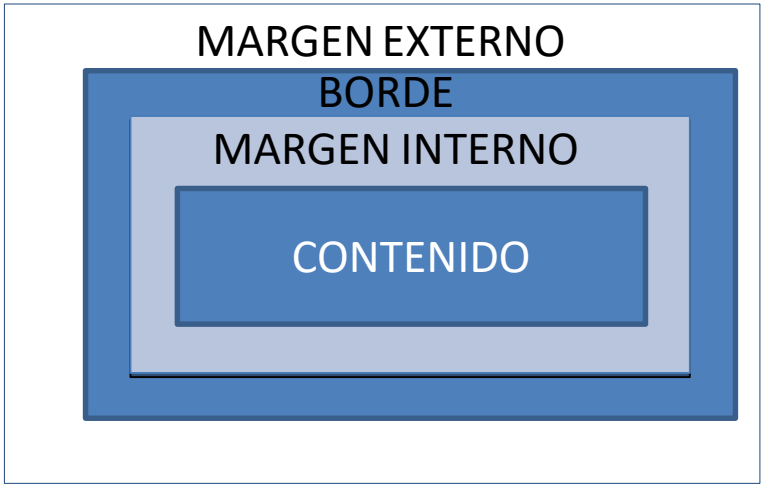
Confucio

15. MODELO DE CAJAS

- La maquetación en CSS se basa en el “modelo de cajas”.
- Según este concepto, todos los elementos se encuentran ubicados en un contenedor rectangular denominado **caja**.
- Estas cajas tienen unas propiedades básicas que afectan a su ubicación en la página y al espacio que ocupan:
  - Margen exterior: **margin**, **margin-top**, **margin-right, margin-bottom**, **margin-left**.
  - Margen interior: **padding**, **padding-top**, **padding-right, padding-bottom**, **padding-left**.
  - Borde: **border**, **border-width**, **border-style** (*obligatoria*), **border-color**. **border-block**. Ocupa espacio, aumenta el tamaño del elemento.
  - Contorno: **outline**, **outline-color**, **outline-width**, **outline-style**. No ocupa espacio, se dibuja encima del elemento.
  - Ancho: **width**.
  - Alto: **height**.

15. MODELO DE CAJAS

- La maquetación en CSS se basa en el “modelo de cajas”.



width y **height** especifican las dimensiones del contenido sin tener en cuenta borde y márgenes.

margin:

- un solo valor: aplica a los cuatro lados,
- dos valores: el primero se aplica a superior e inferior y el segundo a laterales,
- tres valores: el primero al superior, el segundo a los laterales y el tercero al inferior,
- cuatro valores: se aplica a superior, derecho, inferior, izquierdo.

15. MODELO DE CAJAS

FORMATO	PROPIEDAD	VALORES
Color del borde	border-color border-top-color border-bottom-color border-right-color border-left-color	Color en alguna de las notaciones especificadas | transparent
Grueso del borde	border-width Border-top-width Border-bottom-width Border-right-width Border-left-width	Valor de longitud | thin | médium | thick
Ancho del margen interno	padding padding-top padding-bottom padding-right padding-left	Auto | valor de longitud | valor de porcentaje
Estilo del borde	border-style border-top-style border-bottom-style border-right-style border-left-style	Solid | dashed | dotted | double | ridge | Groove | inset | outset | hidden | none
Escritura abreviada del borde	border p.e: border: 1px solid black;	Color Estilo ancho

15. MODELO DE CAJAS

- Hay dos tipos de cajas:
  - Cajas en bloque:
    - Después de la caja, un salto de línea.
    - Suele ocupar el 100% del espacio del contenedor.
    - Se aplican los valores de width y height.
    - Márgenes, bordes y rellenos desplazan a otras cajas.
  - Cajas en línea:
    - No generan salto de línea.
    - No se aplican los valores de width y height.
    - Márgenes, bordes y rellenos horizontales desplazan al resto de cajas en línea.
    - Márgenes, bordes y rellanos verticales NO desplazan al resto de cajas en línea.

15. MODELO DE CAJAS

- Hay dos tipos de cajas:

```
<div>Caja de tipo bloque</div>
<div>Caja de tipo <span>en línea</span> dentro de una caja de tipo bloque</
div>
```

Caja de tipo bloque
Caja de tipo en línea dentro de una caja de tipo bloque

- Todos los elementos HTML pertenecen, por defecto, a uno de estos grupos.
- Esto puede modificarse mediante la propiedad **display**.
- Admite varios valores. Se pueden clasificar en visualización externa e interna. Valores visualización externa:
  - inline: se comporta como un elemento en línea.
  - block: se comporta como un elemento en bloque.
  - none: desactiva todas las modificaciones para el elemento y sus descendientes.

15. MODELO DE CAJAS

- Hay dos tipos de cajas:

```
span {
   display: block;
}
```

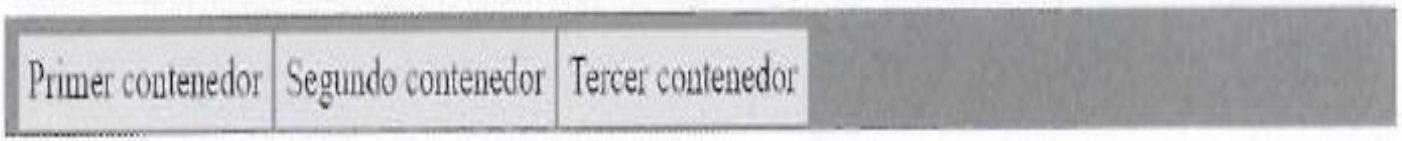
Caja de tipo bloque
Caja de tipo
en línea
dentro de una caja de tipo bloque

- Valores visualización interna:
  - flex: el elemento se comporta como uno de bloque y de acuerdo con el modelo *flexbox*.
    - flex-direction: en qué dirección fluye el contenido del contenedor.
  - grid: el elemento se comporta como uno de bloque y de acuerdo con el modelo *de cuadrícula*.

15. MODELO DE CAJAS. FLEX.

```html
<div id="divFlex">
    <div>Primer contenedor</div>
    <div>Segundo contenedor</div>
    <div>Tercer contenedor</div>
</div>
```

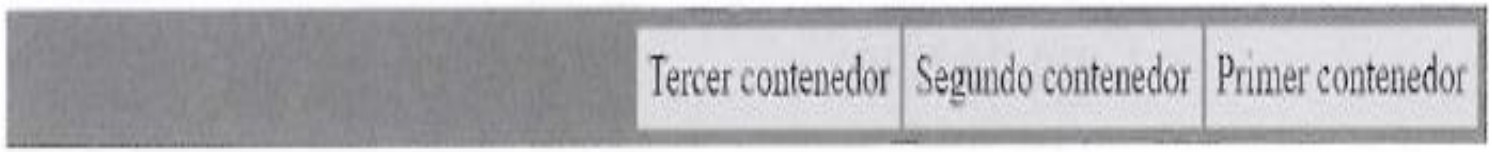
```css
#divFlex {
    display: flex;
    flex-direction: row;
}
```



15. MODELO DE CAJAS. FLEX.

```
<div id="divFlex">
    <div>Primer contenedor</div>
    <div>Segundo contenedor</div>
    <div>Tercer contenedor</div>
</div>
```

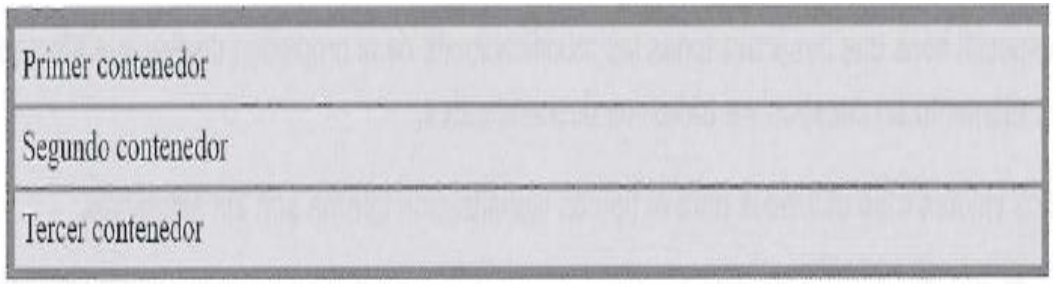
```
#divFlex {
    display: flex;
    flex-direction: row-reverse;
}
```



15. MODELO DE CAJAS. FLEX.

```
<div id="divFlex">
    <div>Primer contenedor</div>
    <div>Segundo contenedor</div>
    <div>Tercer contenedor</div>
</div>
```

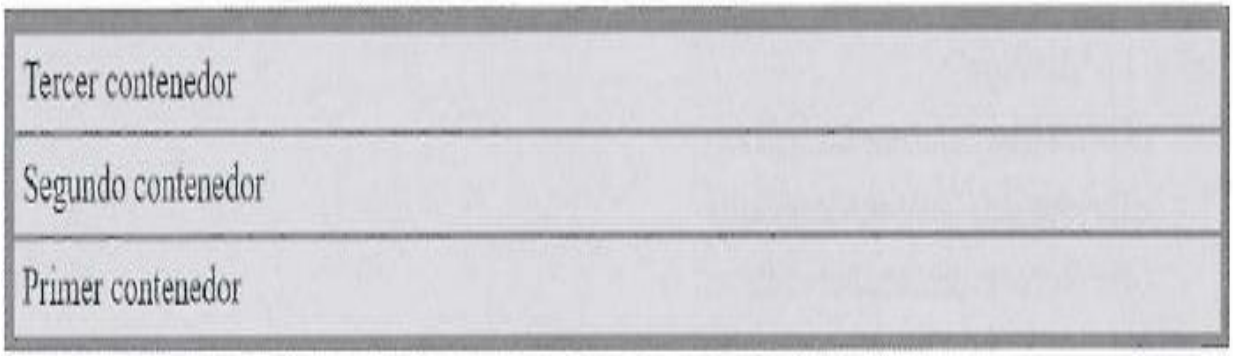
```
#divFlex {
    display: flex;
    flex-direction: column;
}
```



15. MODELO DE CAJAS. FLEX.

```
<div id="divFlex">
    <div>Primer contenedor</div>
    <div>Segundo contenedor</div>
    <div>Tercer contenedor</div>
</div>
```

```
#divFlex {
    display: flex;
    flex-direction: column-reverse;
}
```



15. MODELO DE CAJAS. GRID.

```
<div id="divGrid">
    <div>Primer contenedor</div>
    <div>Segundo contenedor</div>
    <div>Tercer contenedor</div>
    <div>Cuarto contenedor</div>
    <div>Quinto contenedor</div>
    <div>Sexto contenedor</div>
</div>
```

```
#divGrid {
    display:grid;
    grid-template-columns: auto auto auto;
}
```

Primer contenedor	Segundo contenedor	Tercer contenedor
Cuarto contenedor	Quinto contenedor	Sexto contenedor

15. MODELO DE CAJAS. GRID.

```
<div id="divGrid">
    <div>Primer contenedor</div>
    <div>Segundo contenedor</div>
    <div>Tercer contenedor</div>
    <div>Cuarto contenedor</div>
    <div>Quinto contenedor</div>
    <div>Sexto contenedor</div>
</div>
```

```
#divGrid {
    display:grid;
    grid-template-columns: 100px 100px 100px;
}
```

Primer contenedor	Segundo contenedor	Tercer contenedor
Cuarto contenedor	Quinto contenedor	Sexto contenedor

15. MODELO DE CAJAS. GRID.

```
<div id="divGrid">
    <div>Primer contenedor</div>
    <div>Segundo contenedor</div>
    <div>Tercer contenedor</div>
    <div>Cuarto contenedor</div>
    <div>Quinto contenedor</div>
    <div>Sexto contenedor</div>
</div>
```

```
#divGrid {
    display:grid;
    grid-template-columns: 70% 30%;
}
```

Primer contenedor	Segundo contenedor
Tercer contenedor	Cuarto contenedor
Quinto contenedor	Sexto contenedor

16. RESPONSIVE WEB DESGIGN.

- Las páginas web se visualizan utilizando dispositivos diferentes. Tanto la apariencia como el uso, debe ser bueno y fácil independientemente del dispositivo en que se visualice.
- Diseño *responsive,* nuestra página se ve bien en:
  - ordenador de sobremesa,
  - tablet,
  - teléfono móvil.
- Viewport: área visible para el usuario de una página web.
  - Varía según el dispositivo
  - <meta name="viewport" content="width=device-width, initial-scale=1.0">
- Propiedad box-sizing: border-box
  - Recomendable aplicar a todos los elementos.
  - Incluye padding y border dentro del total width y height de los elementos.

```
* {
    box-sizing: border-box;
}
```

16. RESPONSIVE WEB DESGIGN.

- Media Query
  - Usa la norma *@media* para incluir un bloque de CSS que solo tendrá efecto si cierta condición se cumple.

```
@media only screen and (max-width: 600px) {
     body {
               background-color: lightblue;
                                 }
}
```

  - Se pueden añadir todos los @media que se necesiten.

```
@media only screen and (min-width: 400px) {
     /* Móvil: */
          .col-s-1 {width: 8.33%;}
          .col-s-2 {width: 16.66%;}
     }

@media only screen and (min-width: 800px) {
     /* Tablet: */
          .col-1 {width: 8.33%;}
          .col-2 {width: 16.66%;}
     }
```

16. RESPONSIVE WEB DESGIGN.

- Media Query
  - Importante controlar los conflictos.
  - Es posible que haga falta quitar formatos aplicados en otros puntos de la hoja CSS.
  - Para ello, seleccionar aquellos elementos sobre los que quitar formato, y en las propiedades a quitar aplicar valores (depende de la propiedad):
    - unset
    - none
    - …

17. VISUALIZACIÓN DE ELEMENTOS.

- Posicionamiento
  - Permite posicionar los elementos HTML mediante la propiedad **position**.
  - Un elemento HTML puede posicionarse de 5 modos diferentes:
    - Estático
      - position: **static**.
      - No permite mover el elemento, se queda situado en la posición que determine el navegador según el orden de carga.
    - Relativo
      - position: **relative**.
      - Posición relativa a la posición del elemento padre, o, en caso de no existir, de la ventana del navegador.

17. VISUALIZACIÓN DE ELEMENTOS.

- Absoluto
  - position: **absolute**.
  - Se posiciona a partir de su antecesor más cercano (diferencia con *fixed*).
  - Si no tiene antecesor se posiciona en referencia al *body* de la página y se mueve al hacer scroll.
  - Elementos posicionados de manera absoluta siguen un orden distinto al flujo normal y pueden superponerse.
- Fijo
  - position: **fixed**.
  - El elemento es totalmente independiente del resto del documento.
  - Se posiciona a partir de la parte superior izquierda de la ventana del navegador.
  - Permanece en la misma posición aunque el documento se desplace mediante la barra de scroll.

17. VISUALIZACIÓN DE ELEMENTOS.

- Pegajoso
  - position: **sticky**.
  - El elemento se posiciona en base a la posición de scroll del usuario.
  - Es un posicionamiento que se encuentra entre el modo *relativo* y el *fijo*, dependiendo de la posición del scroll. Está posicionado como relativo hasta que se alcanza un desplazamiento de scroll en la ventana gráfica, entonces se pega en un sitio (como si fuera fijo).

17. VISUALIZACIÓN DE ELEMENTOS.

Valor	Descripción
static	Es el valor por defecto. El elemento se posiciona siguiendo el flujo normal del documento. No le afectan las propiedades **top, bottom, left, right** y **z-index.**
relative	El elemento se posiciona siguiendo el flujo normal del documento, pero se puede desplazar de forma relativa a su posición por defecto utilizando las propiedades **top, bottom, left** y **right.** El desplazamiento no desplaza a los demás elementos, por lo que puede proporcionar apilamiento que deberá organizarse utilizando la propiedad **z-index.**
absolute	El elemento deja de seguir el flujo normal del documento. Mediante las propiedades **top, bottom, left** y **right** el elemento se posiciona de manera relativa a su ancestro. Si no tiene ancestro, se posiciona de manera relativa a la página completa.
fixed	El elemento deja de seguir el flujo normal del documento. Mediante las propiedades **top, bottom, left** y **right** el elemento se posiciona de manera relativa a la página completa.
sticky	El elemento se posiciona siguiendo el flujo normal del documento. Mediante las propiedades **top, bottom, left** y **right** se fijan unas posiciones límites relativas a su contenedor, de tal manera que no excederá de estas.

17. VISUALIZACIÓN DE ELEMENTOS.

- Además de elegir el tipo de posicionamiento, debemos marcar la posición del elemento mediante alguna de las propiedades:
  - left: indica la distancia al borde izquierdo del elemento padre o del borde de la ventana del navegador;
  - right: indica la distancia al borde derecho del elemento padre o del borde de la ventana del navegador;
  - bottom: indica la distancia al borde inferior del elemento padre o de la ventana del navegador.
  - top: indica la distancia al borde superior del elemento padre o de la ventana del navegador.
  - Pueden especificarse con unidades de medida o con %.
- EJEMPLOS

18. OTRAS PROPIEDADES VISUALES.

FORMATO	PROPIEDAD	VALORES
Situar un elemento lo más a la derecha o izquierda de su elemento padre.	float	right | left | none
Eliminar alineamiento en un lateral de los elementos flotantes dentro del elemento padre.	clear	right | left | both | none
Definir lo que debe hacer el navegador cuando el elemento hijo es más grande que el elemento padre.	overflow	hidden (oculta la parte desbordada) | scroll | visible (muestra la parte desbordada)| auto
Posicionar varios elementos superpuestos uno encima de otro.	z-index	Numero (a mayor valor mayor cercanía al observador)

18. OTRAS PROPIEADES VISUALES.

- **float**
  - Especifica si un elemento debe flotar hacia la izquierda, hacia la derecha o no flotar.
  - **Los elementos posicionados ignoran la propiedad float**.
  - Los elementos al lado de un elemento flotante fluirán alrededor de él. Para evitar esto, se utiliza la propiedad **clear** o el truco **clearfix**.

index.html

```
<div id="contenedor1" class="contenedor">Contenido del primer contenedor</div>
<div id="contenedor2" class="contenedor">Contenido del segundo contenedor</div>
<div id="contenedor3" class="contenedor">Contenido del tercer contenedor</div>
```

18. OTRAS PROPIEADES VISUALES.

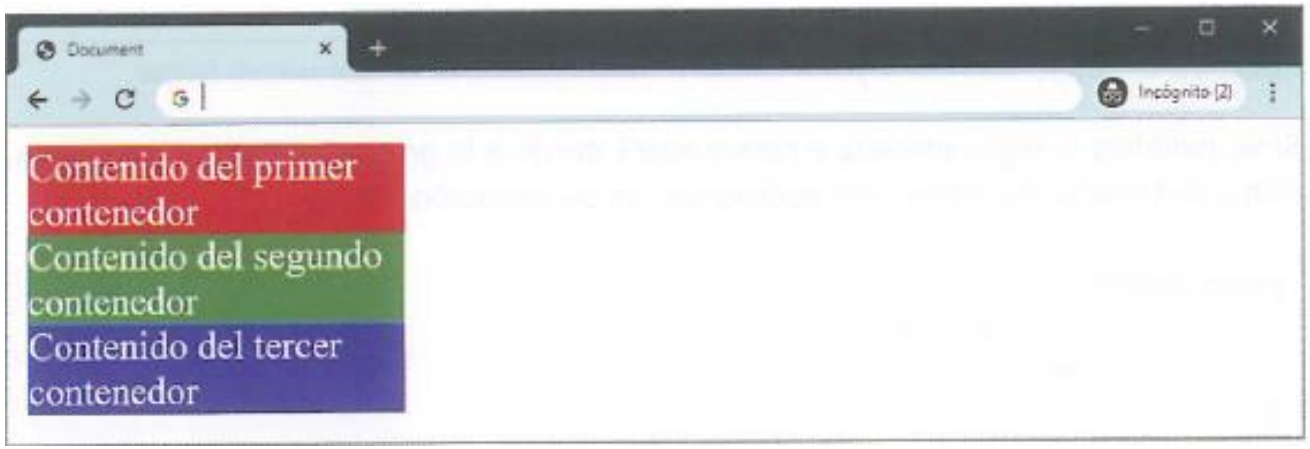
estilos.css

```
.contenedor {
    color:white;
}
#contenedor1 {
    background-color: red;
}
#contenedor2 {
    background-color: green;
}
#contenedor3 {
    background-color: blue;
}
```

18. OTRAS PROPIEADES VISUALES.

estilos.css

```
.contenedor {
    color:white;
    width: 30%;
}
```



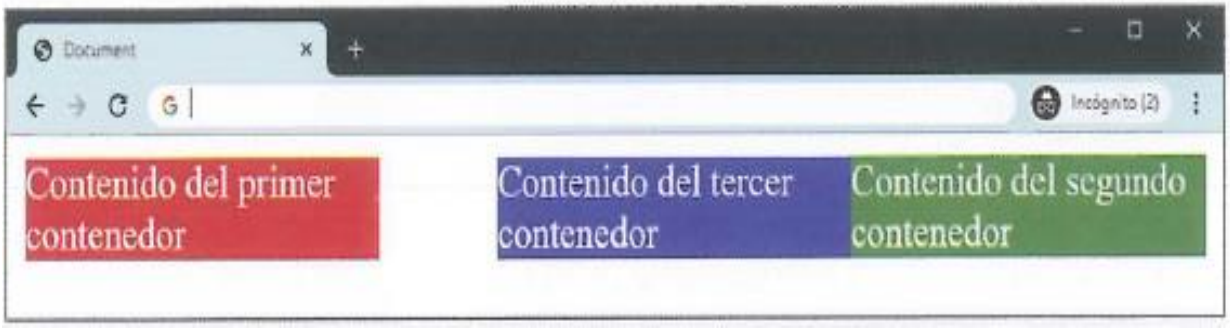
18. OTRAS PROPIEADES VISUALES.

- **float:left:** empuja el elemento hacia el lado izquierdo de su contenedor.
- **float:right:** empuja el elemento hacia el lado derecho de su contenedor.
- **float:none**: no se empuja el elemento. Es el valor por defecto.
- **float:inherit**: hereda el valor de la propiedad **float** del contenedor padre.

estilos.css

```
#contenedor1 {
    background-color: red;
    float:left;
}
#contenedor2 {
    background-color: green;
    float:right;
}
#contenedor3 {
    background-color: blue;
    float:right;
}
```

18. OTRAS PROPIEADES VISUALES.



- Si modificamos el css, diciendo al contenedor3 que no flote, éste e posicionará en su ubicación natural:

estilos.css

```
#contenedor3 {
    background-color: blue;
    float:none;
}
```

18. OTRAS PROPIEADES VISUALES.

- Ver EjemplosFloat en el Aula Virtual.

18. OTRAS PROPIEADES VISUALES.

- **overflow**

Valor	Descripción
visible	Es el valor por defecto. Provoca que el contenido desborde el espacio del contenedor consiguiendo así que sea visible por completo.
hidden	Oculta el contenido que desborda al contenedor.
scroll	Agrega unas barras de desplazamiento *(scroll)* permitiendo así consultar todo el contenido sin que este exceda de los límites del contenedor.
auto	Se comporta igual que el valor **scroll** pero solo si el contenido excede del tamaño del contenedor.

18. OTRAS PROPIEDADES VISUALES.

FORMATO	PROPIEDAD	VALORES
Bordes redondeados	border-radius border-top-right-radius border-top-left-radius border-bottom-right-radius border-bottom-left-radius	Admite de 1 a 4 valores Admite de 1 a 2 valores (borde superior/inferior y borde derecho/izquierdo) (en valor o porcentaje)
Bordes decorados	border-image	Ver especificación
Sombras sobre caja	box-shadow	**x y z color** **x** margen derecho de la sombra **y** margen inferior de la sombra **z** (opcional) intensidad

18. OTRAS PROPIEDADES VISUALES.

FORMATO	PROPIEDAD	VALORES
FORMATO	PROPIEDAD	VALORES
Sombras sobre texto	text-shadow	**x y z color** **x** margen derecho de la sombra **y** margen inferior de la sombra **z** (opcional) intensidad
Nivel de transparencia	opacity	Valor entre 1 y 0, con 1 la opacidad es total, con 0 el elemento desaparece (completamente transparente).
Transformaciones sobre un elemento (rotar, zoom, deformación, etc…)	transform	Rotar, inclinar, escalar, …, elementos. Aplica transformaciones en 2D o 3D a un elemento.