Píldoras

Filtro JSXGraph – Aules

Uso y funcionalidades

1. ¿Qué es JSXGraph y para qué sirve en Aules?

¿Qué es JSXGraph?

JSXGraph es una librería JavaScript de código abierto que permite crear construcciones geométricas interactivas, trazar funciones matemáticas, generar gráficos y visualizar datos directamente en el navegador, sin necesidad de instalar ningún software adicional.

En el contexto de Aules (Moodle), se integra a través de un filtro (plugin) que permite insertar estas construcciones en cualquier recurso de texto: páginas, etiquetas, preguntas de cuestionario, etc.

Nota técnica

Implementación: JavaScript puro, sin dependencias externas.
Compatibilidad: Funciona en todos los navegadores modernos (Chrome, Firefox, Safari, Edge).
Tipo de plugin: `filter_jsxgraph` — actúa como filtro de texto, no como módulo de actividad.

¿Para qué tipo de contenidos es útil JSXGraph en Aules?

JSXGraph resulta especialmente útil en las siguientes situaciones:

- Representación gráfica de funciones matemáticas (álgebra, cálculo, estadística).
- Construcciones geométricas interactivas (puntos, rectas, circunferencias, polígonos).
- Visualización de datos dinámicos en actividades y recursos de curso.
- Preguntas de cuestionario con respuesta gráfica (combinado con el tipo de pregunta STACK).
- Materiales educativos interactivos para cualquier área que requiera representación visual.

Consejo de uso

Aunque su uso más habitual es en Matemáticas y Física, JSXGraph puede emplearse en cualquier área donde se necesite comunicar información de forma gráfica e interactiva.

2. Uso y funcionamiento básico

¿Cómo se inserta una construcción JSXGraph en un recurso de Aules?

Para insertar una construcción, sigue estos pasos:

- **1.** Accede al editor de texto del recurso (página, etiqueta, pregunta, etc.).
- **2.** Cambia al modo de edición HTML (editor de código fuente o editor de texto plano).
- **3.** Inserta la etiqueta `<jsxgraph>` con el código JavaScript de la construcción.
- **4.** Guarda el recurso. El filtro procesará la etiqueta y renderizará la construcción.

Ejemplo:

```
<jsxgraph width="500" height="500" box="mybox">

JXG.Options.slider.snapValues = [-5, -2, -1, 0, 1, 2, 5];
JXG.Options.slider.snapValueDistance = 0.2;

const board = JXG.JSXGraph.initBoard(BOARDID, { boundingbox: [-10, 10, 10, -10],
axis: true });

var a = board.create('slider', [[2, -5], [7, -5], [-5, 1, 5]], { name: 'a' });
var b = board.create('slider', [[2, -6], [7, -6], [-5, 0, 5]], { name: 'b' });
var c = board.create('slider', [[2, -7], [7, -7], [-5, 0, 5]], { name: 'c' });

var f = board.create('functiongraph', [(x) => a.Value() * x * x + b.Value() * x +
c.Value()]);

var txt = board.create('text', [-9, -5,
() => JXG.Math.Numerics.generatePolynomialTerm([c.Value(), b.Value(), a.Value()],
2, 'x', 2)
], { fontSize: 18 });

</jsxgraph>
```

Nota técnica

BOARDID: Es una constante gestionada automáticamente por el filtro. No es necesario definirla manualmente.

Editor recomendado: editor de texto plano para evitar que Moodle modifique el código al guardar.

¿Qué atributos admite la etiqueta <jsxgraph>?

La etiqueta acepta los siguientes atributos principales:

Atributo	Valor por defecto	Descripción
`width`	500 (px)	Anchura del tablero en píxeles.
`height`	400 (px)	Altura del tablero en píxeles.
`aspect-ratio`	—	Ratio anchura/altura. Si se usa junto con width sin height, regula la altura automáticamente.
`numberOfBoards`	1	Permite usar múltiples tableros en un mismo bloque.
`box`	—	ID personalizado para el contenedor. Si se omite, se genera automáticamente (BOARDID).
`ext_formulas`	false	Activa la extensión de integración con preguntas de tipo Fórmulas.
`title`	—	Título descriptivo del tablero (accesibilidad).
`description`	—	Descripción del tablero (accesibilidad).

¿Es posible usar múltiples tableros en un mismo bloque?

Sí. Mediante el atributo numberOfBoards se pueden definir varios tableros independientes dentro de un mismo bloque. En ese caso, los IDs de cada tablero se acceden mediante las constantes BOARDID0, BOARDID1, BOARDID2, etc.

```
<jsxgraph width="500" height="300" numberOfBoards="2">
  var b1 = JXG.JSXGraph.initBoard(BOARDID0, { boundingbox: [-5,5,5,-5], axis: true
});
  var b2 = JXG.JSXGraph.initBoard(BOARDID1, { boundingbox: [-3,3,3,-3], axis: true
});
</jsxgraph>
```

Nota técnica

Cuando se usan múltiples tableros, los atributos width, height y description pueden recibir valores separados por espacios, uno por tablero.

3. Integración con otros elementos de Aules. JSXGraph + STACK

¿Puede JSXGraph usarse dentro de preguntas de cuestionario?

Sí. JSXGraph se puede utilizar en dos contextos dentro de los cuestionarios:

a) Como elemento visual estático (ilustración): Insertando el bloque `<jsxgraph>` en el enunciado de cualquier tipo de pregunta para mostrar una construcción como apoyo visual.

b) Como elemento interactivo con respuesta gráfica (Preguntas STACK): Combinando JSXGraph con el tipo de pregunta STACK. Esto permite que el alumno interactúe con la gráfica (por ejemplo, arrastrando un punto) y que la respuesta quede registrada en Moodle.

JSXGraph + STACK

JSXGraph es una biblioteca de visualización interactiva, por sí solo no evalúa respuestas matemáticas. STACK es un tipo de pregunta de Moodle que integra el sistema de algebra computacional Maxima para evaluar respuestas matemáticas de forma simbólica.

STACK permite incrustar JSXGraph dentro de la pregunta y además **enlazar los valores del gráfico con la respuesta del estudiante**, por ejemplo:

- El estudiante mueve un punto en el gráfico → esa coordenada se envía como respuesta → Maxima la evalúa.

Esto da lugar a preguntas donde la respuesta *es* la interacción gráfica, algo imposible con JSXGraph solo.

Característica	Descripción
JSXGraph solo	Visualización interactiva. No evalúa respuestas ni asigna puntuación.
STACK	evaluación simbólica con Maxima. Feedback automático personalizado.
STACK + JSXGraph	Lo mejor de ambos: grafico interactivo + evaluación matemática real.
Aleatorización	STACK genera parámetros aleatorios con Maxima para cada estudiante.
Dificultad	JSXGraph es más sencillo. STACK requiere aprender sintaxis Maxima.

> **recomendación:** Usa JSXGraph solo para contenido exploratorio sin evaluación. Usa STACK cuando necesites calificar automáticamente.

Ejemplo 1: Pendiente de una recta

El estudiante ve una recta aleatoria y debe escribir su pendiente numéricamente.

Variables de la pregunta

```
m: rand_with_prohib(-5, 5, [0]);  /* pendiente aleatoria, nunca 0 */
b: rand_with_prohib(-4, 4, [0]);  /* intercepto aleatorio */
```

Enunciado de la pregunta (fragmento clave)

```
<p>Observa la siguiente recta y determina su
<strong>pendiente</strong>.</p>
```

```
<div style="display:flex; justify-content:center;">
[[jsxgraph width="400px" height="400px"]]
var board = JXG.JSXGraph.initBoard(BOARDID, {
    boundingbox: [-8, 8, 8, -8],
    axis: true,
    showCopyright: false
});

var m = {#m#};
var b = {#b#};

board.create('functiongraph', [function(x){ return m*x + b; }], {
    strokeColor: '#c00',
    strokeWidth: 2
});

board.create('point', [0, b], {
    name:'(0, '+b+')',
    fixed: true,
    color: 'blue'
});

board.create('point', [1, m+b], {
    name:'(1, '+(m+b)+')',
    fixed: true,
    color: 'blue'
});
[[/jsxgraph]]
</div>

<p>Escribe la pendiente de la recta: [[input:ans1]]
[[validation:ans1]]</p>
```

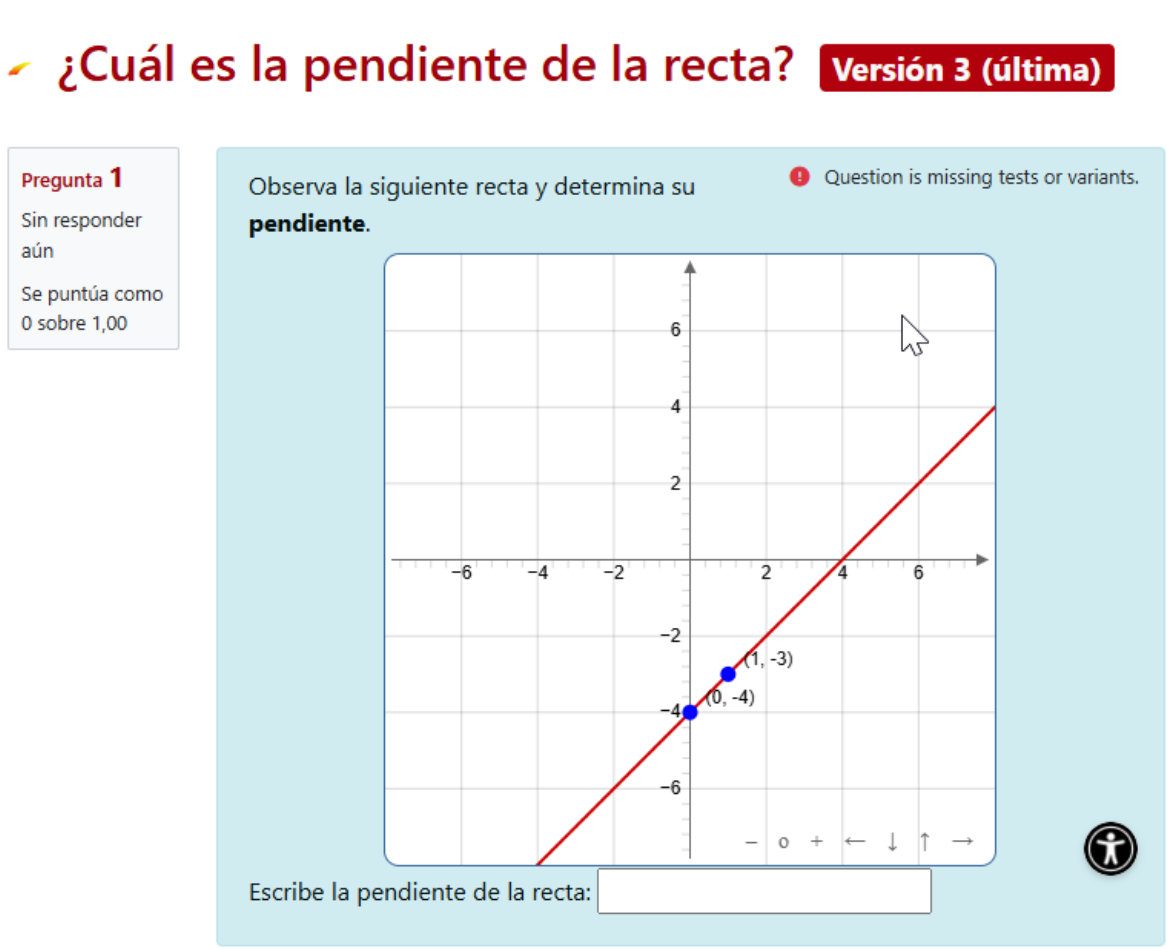
Nota: {#m#} inyecta el valor de Maxima en JavaScript. {@m@} lo muestra en el texto del feedback.

Configuración de la Entrada: ans1

Campo	Valor
Tipo de entrada	Algebraica
Respuesta modelo	m
Prohibir flotantes	Sí

Árbol de respuestas potenciales: prt1 - Nodo 1

Campo	Valor
`Prueba`	AlgEquiv
`TAns`	m
`Si la prueba es verdadera`	1 | Retroalimentación: ¡Correcto! La pendiente es {@m@}.
`Si la prueba es falsa`	0 | Retroalimentación: Recuerda que la pendiente es el coeficiente que acompaña a \(x\). Intenta de nuevo.



Ejemplo 2: Arrastrar el punto a la raíz de una parábola

El estudiante arrastra un punto sobre el eje X hasta donde cree que esta la raíz. JSXGraph envía el valor automáticamente a STACK sin que el estudiante tenga que escribir nada.

Variables de la pregunta

```
a: 1;
r1: rand_with_prohib(-4, 4, [0]);   /* raíz aleatoria */
b: -r1;                              /* y = x² - r1*x  →  raíz en
x=r1 */
c: 0;
```

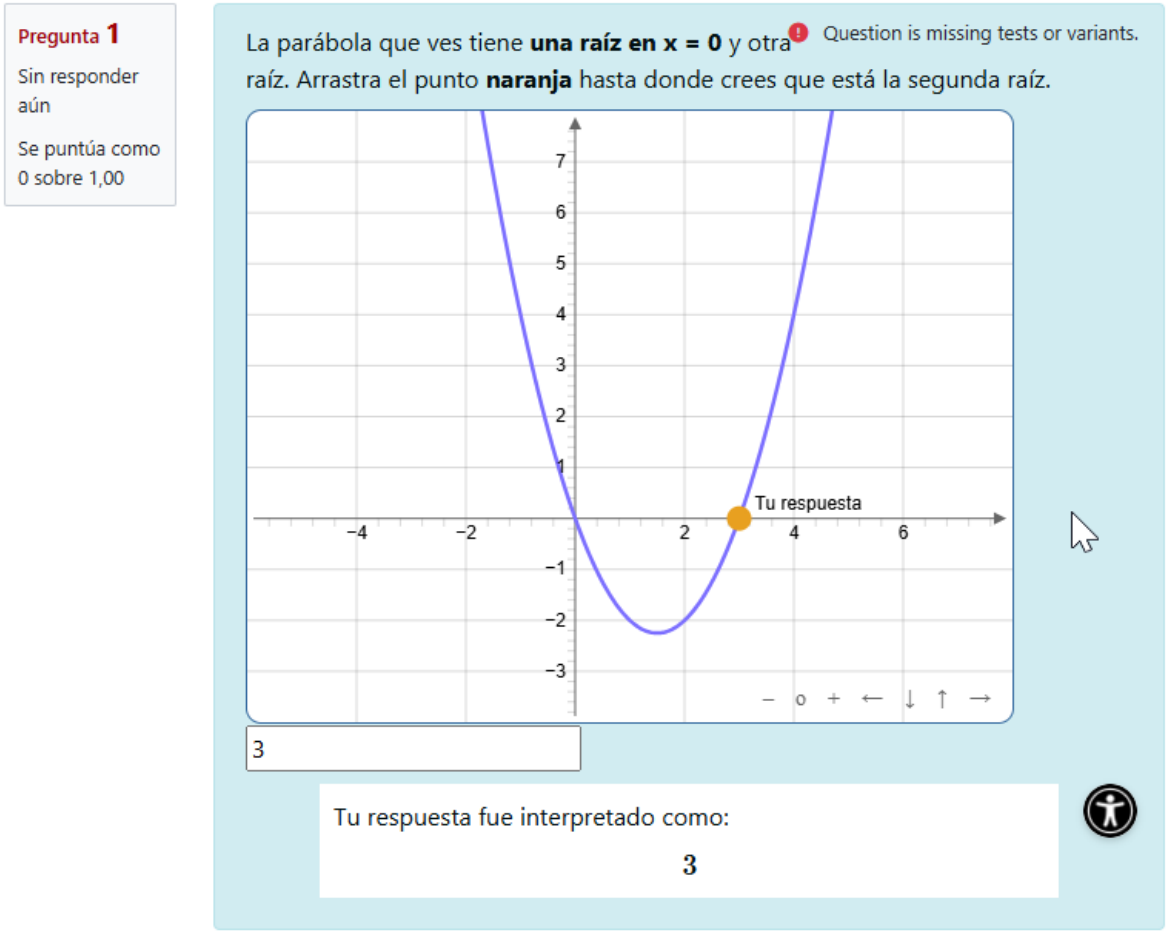
Enunciado de la pregunta (Patrón clave)

Este es el patrón fundamental para enviar valores gráficos a STACK:

```
<p>La parábola que ves tiene <strong>una raíz en x = 0</strong> y
otra raíz. Arrastra el punto <strong>naranja</strong> hasta donde
crees que está la segunda raíz.</p>

[[jsxgraph input-ref-ans1="ans1Ref" width="500px" height="400px"]]

var board = JXG.JSXGraph.initBoard(BOARDID, {
    boundingbox: [-6, 8, 8, -4],
    axis: true,
    showCopyright: false
});

var a = {#a#};
var b = {#b#};
var c = {#c#};

board.create('functiongraph', [function(x){
    return a*x*x + b*x + c;
}], { strokeColor: '#7c6fff', strokeWidth: 2.5 });

var p = board.create('point', [1, 0], {
    name: 'Tu respuesta',
    color: '#e8a020',
    size: 6,
    snapToGrid: true,
    snapSizeX: 1,
    snapSizeY: 1
});

p.Y = function(){ return 0; };
p.update();

/* Escuchar el arrastre directamente sobre el punto */
p.on('drag', function(){
    var val = Math.round(p.X());
    var input = document.getElementById(ans1Ref);
    if(input){
        input.value = val;
        /* Disparar evento de cambio para que STACK detecte el nuevo
valor */
        input.dispatchEvent(new Event('change'));
        input.dispatchEvent(new Event('input'));
    }
});

[[/jsxgraph]]

<p>[[input:ans1]] [[validation:ans1]]</p>
```

Arrastra el punto hasta la raíz de la parábola

Versión 6 (última)



¿Por qué este patrón y no otros?:

- **input-ref-ans1="ans1Ref"** en la etiqueta: hace que STACK resuelva automáticamente el ID real del input, que cambia con cada versión de la pregunta.
- **p.on('drag')** en lugar de board.on('update'): escucha directamente el arrastre del punto, que es más fiable.
- **dispatchEvent('change')** y **dispatchEvent('input'):** notifican a STACK del nuevo valor para que lo procese correctamente.
- Nunca usar el ID hardcodeado (ej: q965578:1_ans1): ese ID cambia con cada pregunta y versión.

Configuración de la Entrada: ans1

Campo	Valor
`Tipo de entrada`	Numerical
`Respuesta modelo`	r1
`Extra options`	mindp:0, maxdp:0

Árbol de respuestas potenciales: prt1 - Nodo 1

Campo	Valor
`Prueba`	NumAbsolute
`TAns`	r1
`Tolerancia`	0.6
`Si la prueba es verdadera`	1 | ¡Correcto! La raíz estaba en x = {@r1@}.
`Si la prueba es falsa`	0 | No es correcto. La raíz estaba en x = {@r1@}.

---

4. Referencia rápida de sintaxis

Inyección de valores Maxima en JSXGraph

Sintaxis	Uso
`{#variable#}`	Inyecta el valor numérico de Maxima en el código JavaScript de JSXGraph.
`{@variable@}`	Muestra el valor calculado de Maxima en el texto HTML (enunciado o feedback).

Funciones Maxima más útiles en STACK

Función	Descripción
`rand_with_prohib(min, max, [excluidos])`	Entero aleatorio excluyendo valores no deseados.
`rand(n)`	Entero aleatorio entre 0 y n-1.
`diff(f, x)`	Derivada de f respecto a x.
`solve([ec], [var])`	Resuelve una ecuación.
`ev(expr, x=valor)`	Evalúa una expresión en un punto.

Tests de respuesta PRT más comunes

Test	Cuándo usarlo
`AlgEquiv`	Respuestas algebraicas: comprueba equivalencia simbólica (ej: x+1 = 1+x).
`NumAbsolute`	Respuestas numéricas con tolerancia absoluta (ej: 3 con tolerancia 0.5).
`NumRelative`	Respuestas numéricas con tolerancia relativa (porcentaje).

Test	Cuándo usarlo
`EqualComAss`	Igualdad conmutativa y asociativa (útil para sumas y productos).

5. Problemas frecuentes y soluciones

La construcción no se muestra después de guardar el recurso. ¿Qué puede estar pasando?

Este es el problema más habitual. Las causas más frecuentes son:

- **Editor incorrecto:** El editor de texto enriquecido de Moodle (TinyMCE) puede eliminar o modificar la etiqueta `<jsxgraph>` al guardar. Solución: cambiar al editor de texto plano antes de insertar el código, entrar al campo 'Enunciado de la pregunta' y cambiar a Formato HTML.
- **Filtro inactivo:** El filtro JSXGraph puede no estar habilitado.
- **Etiqueta malformada:** Comprobar que la etiqueta de apertura y cierre son correctas: `<jsxgraph>` … `</jsxgraph>`.

> **Nota técnica**
> Si tienes configurado TinyMCE como editor por defecto y necesitas insertar código JSXGraph, cambia temporalmente al editor Atto o al editor de texto plano en tus preferencias de usuario (Panel de control → Preferencias → Preferencias del editor).

¿Por qué el tablero se muestra pero aparece en blanco (sin contenido)?

Si el contenedor (div) se renderiza, pero la construcción está vacía, las causas habituales son:

- **Error de JavaScript:** Abrir la consola del navegador (F12 → Consola) para detectar errores de sintaxis en el código.
- **BOARDID incorrecto:** Si se usa un ID personalizado mediante el atributo `box`, asegurarse de referenciarlo correctamente en el código JS.
- **Conflicto de IDs:** Si hay varios tableros en la misma página con el mismo ID, solo se inicializará el primero. Usar `numberOfBoards` o dejar que el filtro genere los IDs automáticamente.

> **Nota técnica**
> **Depuración rápida:** La consola del navegador es la herramienta principal. Cualquier error en el código JSXGraph aparecerá ahí con su línea exacta.

¿Cómo se gestiona la responsividad del tablero en diferentes dispositivos?

Para que el tablero se adapte al tamaño de pantalla, es recomendable usar el atributo aspect-ratio en lugar de definir width y height fijos:

```
<jsxgraph width="100%" aspect-ratio="16/9">
  var board = JXG.JSXGraph.initBoard(BOARDID, {
    boundingbox: [-5, 5, 5, -5],
    axis: true
  });
</jsxgraph>
```

Si lo que se quiere es centrar el gráfico simplemente hay que encapsular el código JSXGraph en un contenedor (div):

```
<div style="display:flex; justify-content:center;">
<jsxgraph width="100%" aspect-ratio="16/9">
  var board = JXG.JSXGraph.initBoard(BOARDID, {
    boundingbox: [-5, 5, 5, -5],
    axis: true
  });
</jsxgraph>
</div>
```

Nota técnica

Regla: Si se especifican tanto `width` como `height`, el atributo `aspect-ratio` se ignora.

Recomendación: Para contextos donde el alumno pueda acceder desde móvil o tablet, definir width en porcentaje y controlar la altura mediante aspect-ratio proporciona una experiencia más consistente.

Error de unidades CSS

Error: The width/height of a JSXGraph must use a known CSS-length unit.

Causa: la etiqueta [[jsxgraph]] no incluye la unidad en el ancho o alto.

Solución: añadir **px** explícitamente:

```
[[jsxgraph width="400px" height="400px"]]
```

Error de version de Máxima

Error: The version of the STACK-Maxima libraries does not match.

Causa: el plugin STACK fue actualizado pero las librerías de Maxima no. Requiere intervención del administrador.

Solución: Administracion del sitio > Plugins > Tipos de preguntas > STACK > Configuración > Update STACK Maxima libraries. Actualizar.

El código JSXGraph funciona en páginas, pero no en preguntas de cuestionario. ¿Por qué?

Este es un comportamiento esperado. La previsualización durante la edición de una pregunta no renderiza el código JSXGraph, pero sí funciona correctamente cuando:

- Guardas la pregunta y la previsualizas desde el banco de preguntas.
- Añades la pregunta a un cuestionario y lo previsualizas como estudiante.
- Los estudiantes realizan el cuestionario.

Pasos para verificar que el código funciona:

1. Edita la pregunta e inserta el código JSXGraph en el enunciado (usando editor Atto o texto plano).
2. Guarda los cambios.
3. Ve al banco de preguntas del curso.
4. Localiza la pregunta y haz clic en el icono de "Vista previa" (ojo 👁).
5. La construcción JSXGraph se mostrará correctamente en la ventana emergente.

He insertado código JavaScript con operadores de comparación (<, >) y JSXGraph no funciona. ¿Qué está pasando?

El problema más habitual es que Moodle interpreta los símbolos < y > como etiquetas HTML y puede modificarlos o eliminarlos al guardar. Esto rompe el código JavaScript.

Ejemplo de código problemático:

```
<jsxgraph width="500px" height="400px">
  var board = JXG.JSXGraph.initBoard(BOARDID, {boundingbox: [-5,5,5,-
5]});
  for (var k = 0; k < 5; k++) {  // ❌ El símbolo < causa problemas
    board.create('point', [k, k*k]);
  }
</jsxgraph>
```

Solución: Sustituir los operadores ‘<’ y ‘>’ por sus entidades HTML equivalentes:

- `<` → `<`
- `>` → `>`
- `<=` → `&le;`
- `>=` → `&ge;`

Ejemplo de código corregido:

```
<jsxgraph width="500px" height="400px">
  var board = JXG.JSXGraph.initBoard(BOARDID, {boundingbox: [-5,5,5,-
5]});
  for (var k = 0; k &lt; 5; k++) {  // ✅ Usa &lt; en lugar de <
    board.create('point', [k, k*k]);
  }
</jsxgraph>
```

> **Nota técnica**
> Otras entidades HTML útiles en código JavaScript:
> • & → &
> • " → "
> • ' → ' (aunque generalmente no es necesario)

6. Recursos y referencias

¿Dónde puedo encontrar ejemplos y documentación adicional?

Los recursos oficiales y más útiles para trabajar con JSXGraph en Aules son:

- **Repositorio oficial del plugin (GitHub):** `https://github.com/jsxgraph/moodle-filter_jsxgraph`
- **Biblioteca de ejemplos exportables:** `https://jsxgraph.org/share` — permite exportar directamente al formato del filtro Moodle.
- **Documentación de la API de JSXGraph:** `https://jsxgraph.org/docs/`
- **Integración con preguntas Fórmulas:** `https://github.com/jsxgraph/moodleformulas_jsxgraph`

> **Consejo de productividad**
> Los ejemplos de jsxgraph.org/share incluyen la opción de exportar directamente en formato de etiqueta <jsxgraph> lista para pegar en Moodle, lo que permite reutilizar construcciones sin necesidad de escribir el código desde cero.