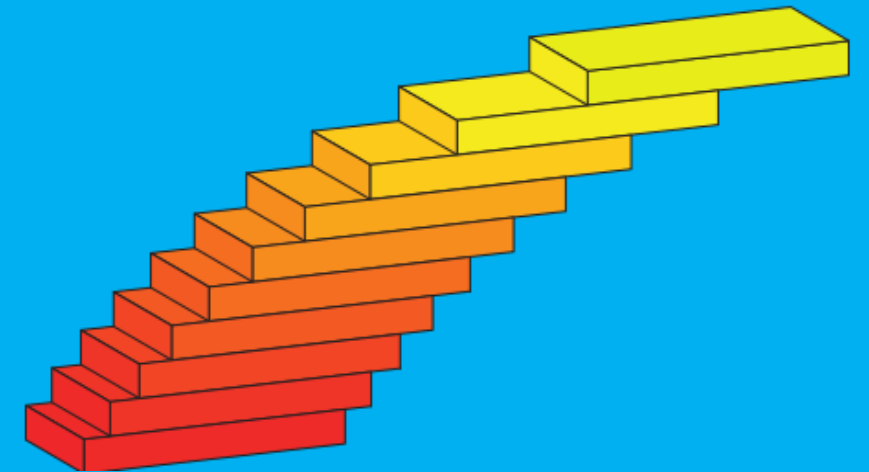


# Parametrisierbare Aufgaben mit STACK: Interaktive Grafiken mit JSXGraph



*Stand: 02.10.2026*

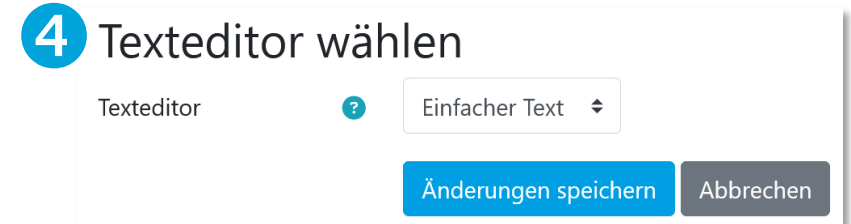
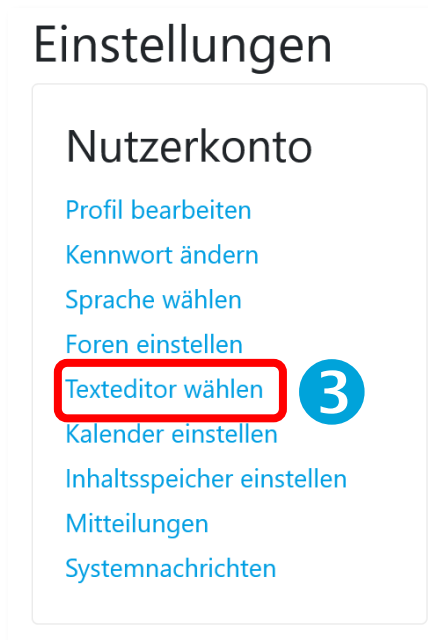
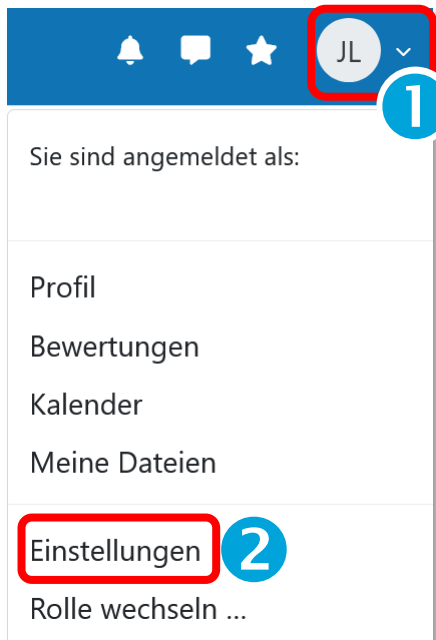
# STACK // Einführung

- In dieser Anleitung wird erklärt, wie STACK-Aufgaben erstellt werden können, in deren Fragetext interaktive Grafiken eingebunden sind.
- In einer solchen Grafik können Studierende Veränderungen vornehmen (z. B. durch das Verschieben von Punkten oder das Verändern von Schieberegler). Dies kann als Antwort der Studierenden genutzt werden.
- STACK verwendet dafür JSXGraph, eine freie Open-Source JavaScript-Bibliothek, die an der Universität Bayreuth entwickelt wird (<https://jsxgraph.uni-bayreuth.de>).
- JSXGraph wird bereits mit STACK mitgeliefert und ist fest in STACK integriert.

*Hinweis: Wir empfehlen, vor dem Anschauen dieser Anleitung zunächst die Klickanleitung „Erstellung einer Aufgabe“ durchzuarbeiten oder das gleichnamige Videotutorial zu schauen.*

# STACK // Änderung des Texteditors

- JSXGraph-Grafiken werden mit der Programmiersprache JavaScript erstellt. Für das Schreiben von JavaScript-Code ist der in Moodle standardmäßig eingestellte Texteditor „TinyMCE“ allerdings nicht geeignet.
- Ändern Sie zu Beginn der Aufgabenerstellung daher den Texteditor in den Moodle-Einstellungen.
- Klicken Sie dazu oben rechts auf **1 Ihr Moodle-Profilfoto oder Ihre Initialen**, dann auf **2 Einstellungen** und schließlich auf **3 Texteditor wählen**. Wählen Sie dann die Option „Einfacher Text“ aus der Liste aus und speichern Sie die Änderungen **4**.



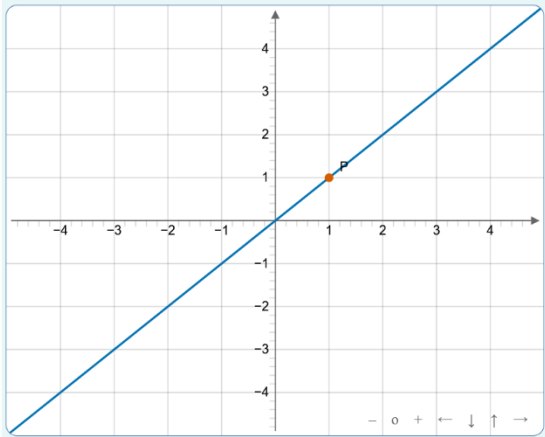
*Hinweis: Der Texteditor wird so bis auf Widerruf geändert. Wenn Sie nach der Aufgabenerstellung für andere Moodle-Inhalte wieder den ursprünglichen Texteditor verwenden möchten, können Sie die Wahl des Texteditor über den gleichen Weg zurück ändern.*

# STACK // Übersicht über Beispielaufgaben

Wie STACK-Aufgaben mit solchen Grafiken erstellt werden können, wird im Folgenden an den folgenden Beispielaufgaben demonstriert:

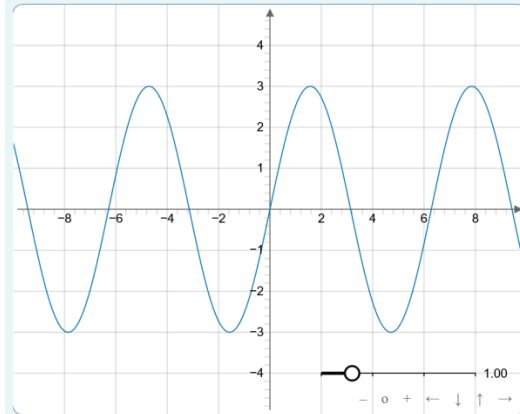
## 1. Ursprungsgerade durch Punkt:

Verschieben Sie den Punkt  $P$  so, dass die Gerade Steigung 2 hat.



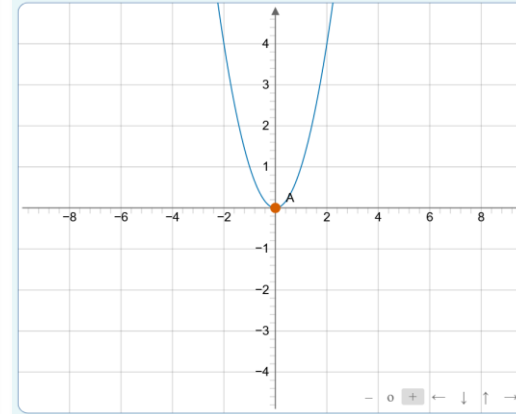
## 2. Sinusfunktion mit Schieberegler:

Setzen Sie den Schieberegler auf einen Wert, sodass die angezeigte Funktion genau für alle  $x \in \mathbb{Z}$  eine Nullstelle hat.



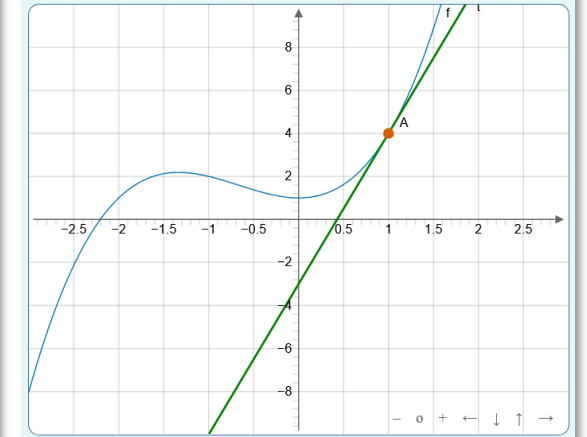
## 3. Parabel verschieben:

Verschieben Sie den Punkt so, dass die Nullstellen der Parabel bei  $x_1 = -2$  und  $x_2 = 2$  liegen.



## 4. Wendetangente markieren:

Verschieben Sie den Punkt  $A$  so, dass  $t$  die Wendetangente von  $f$  ist.

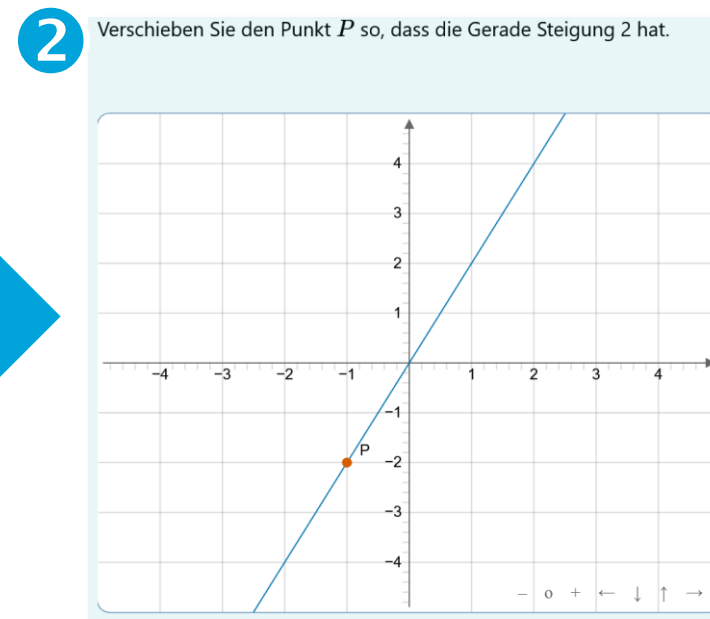
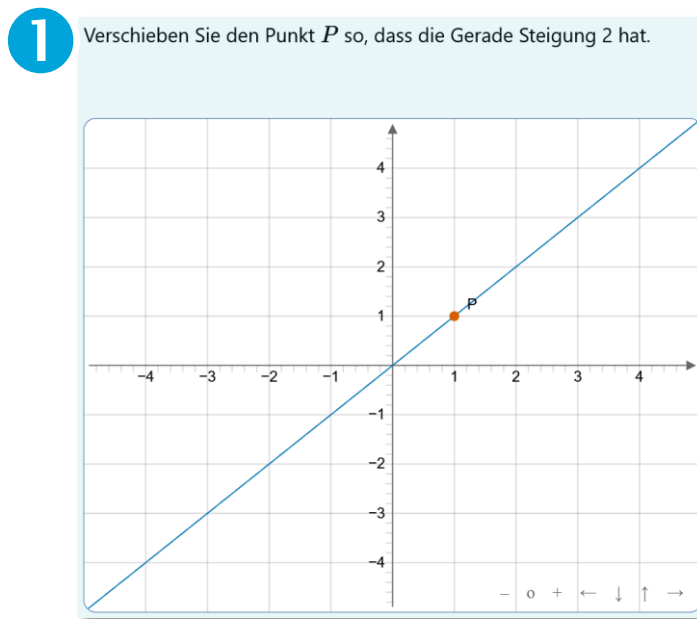


# Aufgabe 1: Ursprungsgerade durch Punkt

*Stand: 02.10.2026*

# STACK // Aufgabe 1: Ursprungsgerade durch Punkt

- In dieser **1. Aufgabe** sind ein verschiebbarer Punkt  $P$  und eine Ursprungsgerade durch  $P$  enthalten. Wenn die Studierenden  $P$  verschieben, ändert sich die Gerade.
- Die Aufgabe ist  $P$  so zu verschieben, dass die Gerade eine Steigung von 2 hat, zum Beispiel **2. so**:



*Hinweis: Insbesondere geben die Studierenden ihre Antwort nicht in ein Eingabefeld ein, sondern antworten ausschließlich durch ihre Veränderungen in der Grafik.*

# STACK // Aufgabe 1: Fragetitel und Fragetext

- Starten Sie mit einer neuen STACK-Aufgabe und geben Sie ihr einen geeigneten Fragetitel.
- Tragen Sie in das Feld „Fragetext“ zunächst den Aufgabentext ein, der den Studierenden angezeigt werden soll.
- Da der „Einfacher Text“-Texteditor von Moodle verwendet wird, muss hier mit HTML gearbeitet werden. Wenn Sie den **1 Aufgabentext** zwischen die vorhandenen `<p>`-Tags einfügen, wird dieser in einem eigenen Absatz angezeigt (eine Übersicht über die wichtigsten HTML-Befehle finden Sie auf der [nächsten Folie](#)), zum Beispiel so:

`<p>Verschieben Sie den Punkt \(\text{P}\) so, dass die Gerade Steigung 2 hat.</p>`

**1** Fragetext  

```
<p>Verschieben Sie den Punkt \(\text{P}\) so, dass die Gerade Steigung 2 hat.</p>
```

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

# STACK // Exkurs: Übersicht über die wichtigsten HTML-Befehle

| HTML-Befehl                                                                                           | Beschreibung                           |
|-------------------------------------------------------------------------------------------------------|----------------------------------------|
| <code>&lt;p&gt;Text&lt;/p&gt;</code>                                                                  | Text erscheint in eigenem Absatz       |
| <code>&lt;br&gt;</code>                                                                               | Zeilenumbruch                          |
| <code>&lt;b&gt;Text&lt;/b&gt;</code>                                                                  | Text wird <b>fett</b> gesetzt          |
| <code>&lt;i&gt;Text&lt;/i&gt;</code>                                                                  | Text wird <i>kursiv</i> gesetzt        |
| <code>&lt;u&gt;Text&lt;/u&gt;</code>                                                                  | Text wird <u>unterstrichen</u>         |
| <code>&lt;del&gt;Text&lt;/del&gt;</code>                                                              | Text wird <del>durchgestrichen</del>   |
| <code>&lt;hr&gt;</code>                                                                               | Horizontale Linie                      |
| <code>&lt;h1&gt;Überschrift&lt;/h1&gt;</code><br>...<br><code>&lt;h6&gt;Überschrift&lt;/h6&gt;</code> | Überschriften auf verschiedenen Ebenen |
| <code>&lt;a href="https://example.org"&gt;Text&lt;/a&gt;</code>                                       | Einfügen eines Hyperlinks              |
| <code>&lt;!-- Kommentar --&gt;</code>                                                                 | Kommentar (Text wird nicht angezeigt)  |
| <code>&lt;span style="color:red"&gt;Text&lt;/span&gt;</code>                                          | Farbiger Text (hier: <b>rot</b> )      |

# STACK // Aufgabe 1: JSXGraph-Platzhalter und Variablen

- Als nächstes wird im Fragetext die Grafik hinzugefügt. Dazu muss JavaScript-Code geschrieben werden. Die Anleitung ist im Folgenden so kleinschrittig, dass Sie keine Programmierkenntnisse benötigen.
- Fügen Sie zunächst die **1 Platzhalter** `[[jsxgraph]]...[[/jsxgraph]]` ein. An dieser Stelle im Fragetext erscheint später die Grafik und zwischen die Platzhalter wird gleich der JavaScript-Code eingetragen.
- Nun wird für jedes Objekt, das angezeigt werden soll, eine JavaScript-Variable erstellt. Für JSXGraph-Grafiken können JavaScript-Variablen mit dem Schlüsselwort `const` definiert werden (s. Hinweis unten).
- Nach `const` folgt der Name der Variable gefolgt von einem Gleichheitszeichen, dem Wert der Variable und einem Semikolon als Abschluss der Zeile:

```
const name = ...;
```

Fragetext



<p>Verschieben Sie den Punkt \(\text{P}\) so, dass die Gerade Steigung 2 hat.</p>

[[jsxgraph]]

1

[[/jsxgraph]]

*Hinweis: Allgemein können Variablen in JavaScript mit den Schlüsselwörtern `let` (wenn der Wert später überschrieben wird), `const` (wenn der Wert später nicht überschrieben wird) und `var` (gilt als veraltet) erstellt werden. Für JSXGraph genügt es in der Regel, `const` zu kennen.*

# STACK // Aufgabe 1: Das Board

- Die erste Variable, die Sie nun anlegen, ist die für das Koordinatensystem („Board“). Schreiben Sie dazu:

```
const board = JXG.JSXGraph.initBoard(divid, {  
  ...  
});
```

- Zwischen die geschweiften Klammern können noch optionale Einstellungen gesetzt werden, zum Beispiel:
  - `boundingbox: [-x, y, x, -y]` → Abschnitte der Achsen des Koordinatensystems
  - `axis: false` → blendet die Achsen und das Koordinatengitter aus (wird hier *nicht* verwendet!)
  - `showCopyright: false` → blendet einen möglicherweise störenden Copyright-Hinweis aus

```
// Board (Koordinatensystem):  
const board = JXG.JSXGraph.initBoard(divid, {  
  boundingbox: [-5, 5, 5, -5],  
  showCopyright: false  
});
```

*Hinweis: Es ist sinnvoll, Zeilenumbrüche und Einrückungen zu verwenden, damit der Code übersichtlich bleibt. Zudem können Sie via // einzeilige und via /\*...\*/ mehrzeilige Kommentare verwenden.*

## STACK // Aufgabe 1: Einen Punkt hinzufügen (1)

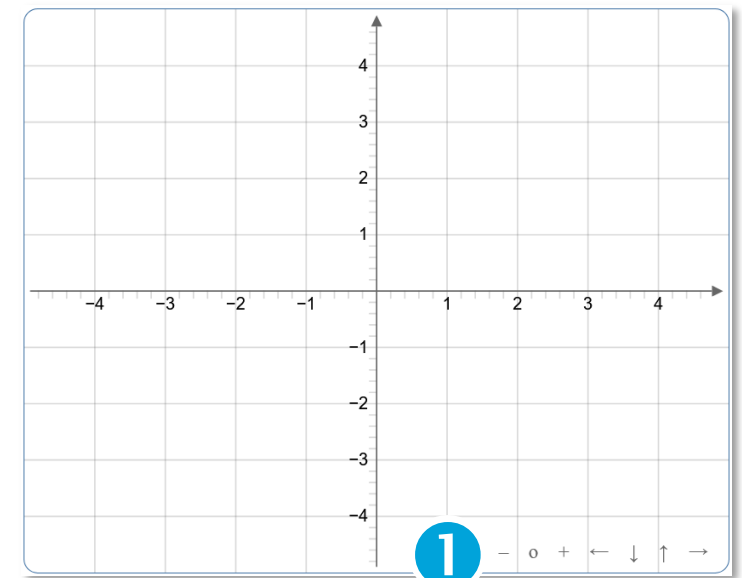
- Wenn Sie Ihre Aufgabe nun speichern, besteht die Grafik aus einem leeren Koordinatensystem mit x- und y-Achsenabschnitten bei -5 bzw. 5. Über die **1 Navigationsleiste** können Sie die Ansicht verschieben sowie hinein- und herauszoomen.
- Als nächstes wird eine Variable für den Punkt benötigt, den die Studierenden verschieben können (Objekttyp point). Legen Sie dazu eine neue Variable p an:

```
const p = board.create("point", [1, 1] {  
  ...  
});
```

Füge dem Board  
ein Objekt hinzu

Objekttyp  
für Punkt

Startkoordinaten  
des Punkts



## STACK // Aufgabe 1: Einen Punkt hinzufügen (2)

- Wieder können zwischen den geschweiften Klammern zusätzliche Einstellungen gesetzt werden. Diese beziehen sich auf das genaue Aussehen und Verhalten des Objekts, zum Beispiel:
  - `size:2` → setzt die Größe des Punkts auf 2
  - `name:"P"` → bewirkt, dass als Name des Punkts „P“ angezeigt wird
  - `snapToGrid:true` → bewirkt, dass der Punkt nicht völlig frei, sondern nur auf einem Raster bewegt werden kann
  - `snapSizeX:0.1` und `snapSizeY:0.1` → bewirken in Kombination mit `snapToGrid:true`, dass der Punkt in x- bzw. y-Richtung mit einer Schrittweite von 0,1 Einheiten verschoben werden kann
- Insgesamt sollte der Code dann so aussehen:

```
const p = board.create("point", [1,1], {  
  size:2,  
  name:"P",  
  snapToGrid:true,  
  snapSizeX:0.1,  
  snapSizeY:0.1  
});
```

## STACK // Aufgabe 1: Eine Gerade durch den Punkt hinzufügen

- Jetzt fehlt nur noch die Gerade durch den Ursprung und  $P$ . Diese kann durch eine weitere Variable  $f$  hinzugefügt werden (Objekttyp `line`):

```
const f = board.create("line", [...]);
```

- Zwischen die eckigen Klammern müssen dann die beiden Punkte eingefügt werden, aus denen die Gerade konstruiert werden soll. Dies sind die Punkte  $[0, 0]$  und  $p$  (Variablenname). Insbesondere können bei der Konstruktion sowohl Koordinaten als auch bestehende Variablen verwendet werden.
- Insgesamt sollte der Code für die Gerade also so aussehen:

```
const f = board.create("line", [[0,0],p]);
```

*Tipp: Auch bei der Gerade können Sie nach Belieben Attribute zwischen geschweifte Klammern setzen, um das genaue Aussehen und Verhalten des Objekts zu beeinflussen, zum Beispiel `strokeColor:"green"` (grüne Linie), `dash:2` (gestrichelte Linie) oder `strokeWidth:3` (dickere Linie).*

# STACK // Aufgabe 1: Verbindung zum Eingabefeld

- In der Grafik sind nun alle nötigen Objekte vorhanden (verschiebbarer Punkt und Gerade durch Punkt).
- Wenn die Studierenden nun den Punkt in der Grafik verschieben würden, würden die Koordinaten des Punktes allerdings noch nicht automatisch in das STACK-Eingabefeld übernommen werden.
- Um eine Verbindung zwischen dem Punkt und dem Eingabefeld ans1 herzustellen, müssen die folgenden Änderungen vorgenommen werden:
  1. Ergänzen Sie im Eröffnungs-`[[jsxgraph]]`-Tag Folgendes: `input-ref-ans1="ans1Ref"` (dabei ist ans1 der Name des Eingabefelds und ans1Ref eine Art Hilfsvariable, deren Name beliebig gewählt werden kann) → `[[jsxgraph input-ref-ans1="ans1Ref"]]`...`[/jsxgraph]`
  2. Ergänzen Sie im JSXGraph-Code unterhalb der Zeile, in der der Punkt p definiert wurde, Folgendes: `stack_jxg.bind_point(ans1Ref,p);` (dabei ist ans1Ref dieselbe Hilfsvariable wie oben und p der Name der Variable, in der der Punkt definiert wurde)

*Hinweis: Die nun eingerichtete Verbindung zwischen dem Punkt und dem Eingabefeld funktioniert in beide Richtungen: Wenn der Punkt bewegt wird, ändern sich die Koordinaten im Eingabefeld. Wenn die Koordinaten im Eingabefeld verändert werden, wird dadurch der Punkt bewegt.*

# STACK // Aufgabe 1: Eingabefeld ausblenden und Bereich „Eingabe“

- Oft ist es bei solchen Aufgaben sinnvoll, das Eingabefeld noch auszublenden, da es nicht benötigt wird und die Studierenden möglicherweise verwirrt.
- Fügen Sie dazu den HTML-`<p>`-Tags im Fragetext, zwischen denen sich die Platzhalter für das Eingabe- und Validierungsfeld befinden, das **1** *Attribut* `style="display:none;"` hinzu:

Fragetext  

**1** `<p style="display:none;">[[input:ans1]] [[validation:ans1]]</p>`

- Erstellen Sie in den **2** *Aufgabenvariablen* eine Variable für eine korrekte Antwort (z. B. `ta1: [1,2];`)
- Verwenden Sie diese Variable im Feld **3** *Musterlösung* im Bereich „Eingabe“.
- Achten Sie im Bereich „Eingabe“ zudem darauf, dass die Verwendung von **4** *Fließkommazahlen* nicht verboten wird.

**2** Aufgabenvariablen 

`ta1: [1,2];`

▼ Eingabe: ans1

Eingabetyp  Algebraische Eingabe

Musterlösung  ta1 **3**

Verbiete Fließkommazahlen  Nein **4**

# STACK // Aufgabe 1: Rückmeldebaum

- Abschließend muss im Rückmeldebaum überprüft werden, ob die Steigung der Gerade, die sich aus der Verschiebung des Punktes durch die Studierenden ergibt, tatsächlich wie gefordert 2 beträgt.
- Legen Sie dazu eine **1 Feedbackvariable** stg für die Steigung an. Der Inhalt des Eingabefelds ans1 entspricht den Koordinaten des Punktes, den die Studierenden verschoben haben. Um daraus die Steigung der Geraden zu bestimmen, wird die y-Koordinate dieses Punkts benötigt, auf die man mit ans1[2] zugreifen kann. Dieser Wert muss durch die x-Koordinate ans1[1] geteilt werden.

**1** Feedback-Variablen ?

- Konfigurieren Sie den ersten Knoten des **2 Rückmeldebaums** dann wie folgt um zu überprüfen, ob die Steigung stg um höchstens 0,1 von der korrekten Steigung 2 abweicht:

- SAns: stg
- TAns: 2
- Antwortüberprüfung: NumAbsolute
- Test-Optionen: 0.1

**2** Knoten 1 ?

Beschreibung

Antwortüberprüfung  SAns

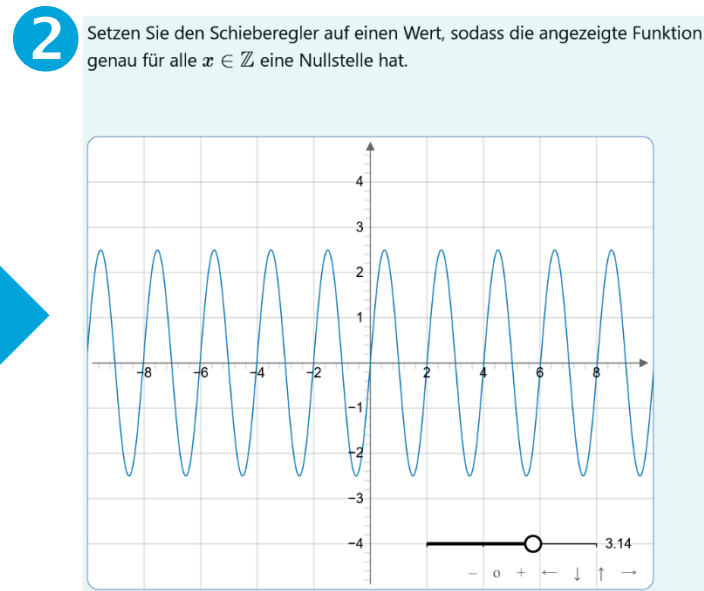
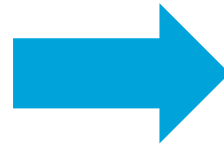
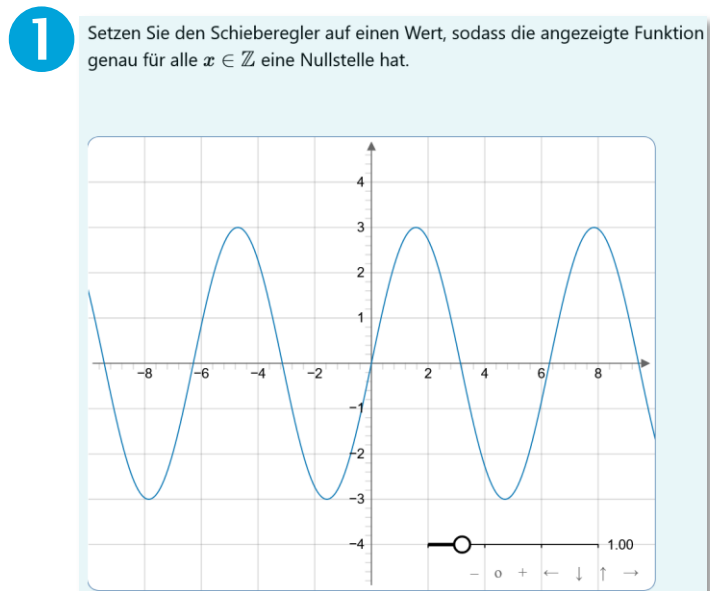
TAns  Test-Optionen  Feedback unterdrücken

# Aufgabe 2: Sinusfunktion mit Schieberegler

*Stand: 02.10.2026*

# STACK // Aufgabe 2: Sinusfunktion mit Schieberegler

- In dieser **1. Aufgabe** ist der Graph einer Sinusfunktion gegeben. Mithilfe eines Schiebereglers kann die Periode der Funktion verändert werden.
- Die Aufgabe ist, den Schieberegler auf einen Wert zu setzen, sodass die Funktion genau für alle ganzen Zahlen eine Nullstelle besitzt. Die korrekte Antwort ist ein Wert nahe  $\pi$  **2**:



## STACK // Aufgabe 2: Fragetitel, Fragetext und Board

- Starten Sie erneut mit einer neuen STACK-Aufgabe und geben Sie ihr einen geeigneten Fragetitel.
- Tragen Sie in das Feld „Fragetext“ zwischen die HTML-`<p>`-Tags wieder den Aufgabentext ein, der den Studierenden angezeigt werden soll, zum Beispiel so:  
`<p>Setzen Sie den Schieberegler auf einen Wert, sodass die angezeigte Funktion genau für alle  $x \in \mathbb{Z}$  eine Nullstelle hat.</p>`
- Fügen Sie nun wieder die Platzhalter für den JSXGraph-Block ein. Die Variable für das Board können Sie aus der vorherigen Aufgabe übernehmen und müssen nur die Achsenabschnitte (`boundingbox: [-10, 5, 10, -5]`) anpassen:

```
[[jsxgraph]]
const board = JXG.JSXGraph.initBoard(divid, {
  boundingbox: [-10, 5, 10, -5],
  axis: true,
  showCopyright: false
});
[/jsxgraph]]
```

## STACK // Aufgabe 2: Schieberegler („Slider“) hinzufügen

Fügen Sie dem Board nun wie folgt einen **1 Schieberegler** hinzu (Objektyp `slider`):

```
const s = board.create("slider", [ [2, -4], [8, -4], [0, 1, 5] ], {  
  snapWidth: 0.01  
});
```

Bewirkt, dass der Slider auf zwei Nachkommastellen genau eingestellt werden kann

Startkoordinaten der Linie des Sliders

Endkoordinaten der Linie des Sliders

Minimaler, voreingestellter und maximaler Wert des Sliders



## STACK // Aufgabe 2: Funktionsgraph hinzufügen

- Fügen Sie dem Board als nächstes den Funktionsgraph hinzu (Objektyp `functiongraph`):

```
const f = board.create("functiongraph", [...]);
```

- Zwischen die eckigen Klammern muss nun die Funktion  $f(x) = \sin(s \cdot x)$  in der JavaScript-Schreibweise eingefügt werden (wobei  $s$  der Wert des Sliders ist). Dies geht wie folgt:

```
const f = board.create("functiongraph", [
  x=>Math.sin(s.Value())*x
]);
```

Sinus-Funktion in JavaScript

Wert des Sliders  $s$

- In Funktionsvorschriften und auch an vielen weiteren Stellen im JSXGraph-Code können auch STACK-Aufgabenvariablen genutzt werden. Soll bspw. die Sinusfunktion noch mit einer randomisierten Zahl (z. B. `a: rand_with_step(2, 4, 0.5);`) multipliziert werden, kann diese Variable via `{#a#}` hinzugefügt werden (insbesondere *nicht* wie sonst `{@...@}`-Platzhalter!):

```
x=>{#a#}*Math.sin(s.Value()*x)
```

## STACK // Aufgabe 2: Binding und Fertigstellung der Aufgabe

- Nun muss noch eine Verbindung zwischen dem Slider und dem Eingabefeld ans1 hergestellt werden. Im Vergleich zu Aufgabe 1 muss in Schritt 2 `bind_slider` statt `bind_point` verwendet werden:
  - Ergänzen Sie im Eröffnungs-`[[jsxgraph]]`-Tag Folgendes: `input-ref-ans1="ans1Ref"` (dabei ist `ans1` der Name des Eingabefelds und `ans1Ref` eine Art Hilfsvariable, deren Name beliebig gewählt werden kann) → `[[jsxgraph input-ref-ans1="ans1Ref"]]`...`[[/jsxgraph]]`
  - Ergänzen Sie im JSXGraph-Code unterhalb der Zeile, in der der Slider `s` definiert wurde, Folgendes: `stack_jxg.bind_slider(ans1Ref, s);` (dabei ist `ans1Ref` dieselbe Hilfsvariable wie oben und `s` der Name der Variable, in der der Slider definiert wurde)
- Blenden Sie erneut das Eingabefeld aus:  
`<p style="display:none;">[[input:ans1]] [[validation:ans1]]</p>`
- Erstellen Sie in den Aufgabenvariablen eine Variable für die korrekte Antwort (z. B. `ta1: 3.14;`), verwenden Sie diese Variable im Feld „Musterlösung“ im Bereich „Eingabe“ und achten Sie wieder darauf, dass die Verwendung von Fließkommazahlen erlaubt ist.
- Konfigurieren Sie Ihren Rückmeldebaum wie im **1 Screenshot** zu sehen.

The screenshot shows the configuration for 'Knoten 1' in the STACK system. It includes a description field, an answer validation dropdown set to 'NumAbsolute', and input fields for 'SAns' (containing 'ans1') and 'TAns' (containing 'ta1'). There are also fields for 'Test-Optionen' (set to '0.1') and a 'Feedback unterdrücken' button with a dropdown set to 'Nein'.

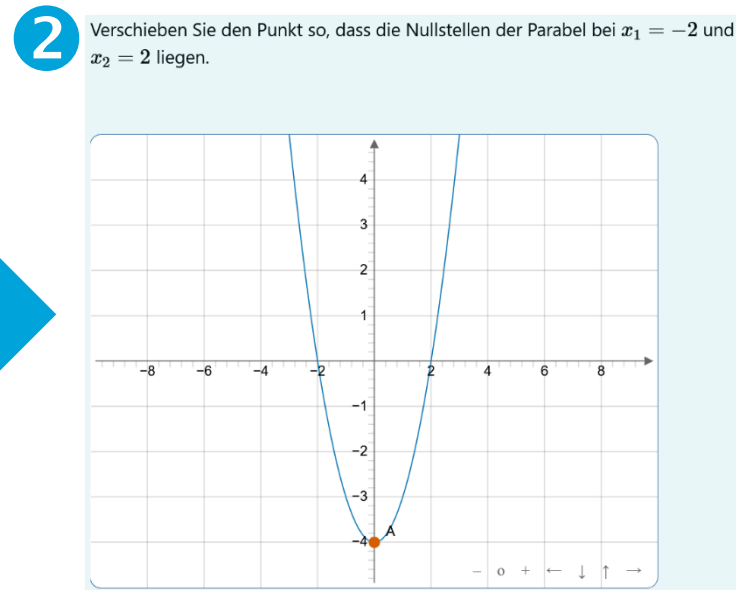
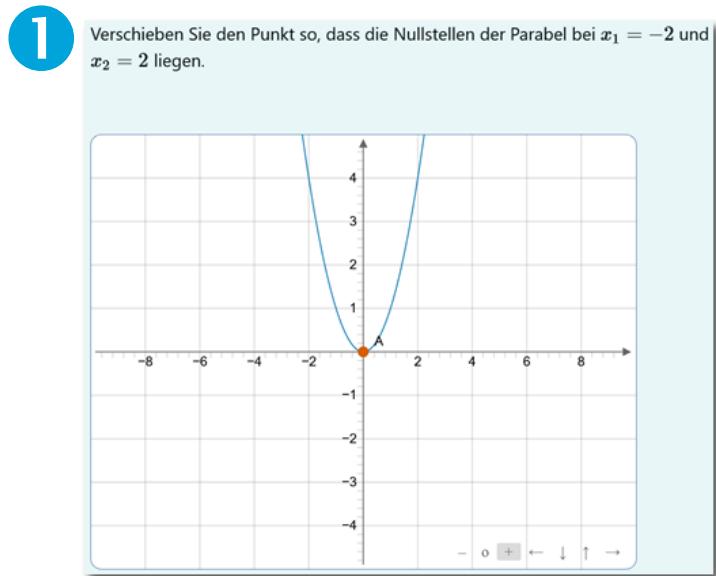
| Knoten 1 |  | Beschreibung         | Antwortüberprüfung          |
|----------|--|----------------------|-----------------------------|
| <b>1</b> |  | <input type="text"/> | NumAbsolute                 |
|          |  | SAns: ans1           | TAns: ta1                   |
|          |  | Test-Optionen: 0.1   | Feedback unterdrücken: Nein |

# Aufgabe 3: Parabel verschieben

*Stand: 02.10.2026*

# STACK // Aufgabe 3: Parabel verschieben

- In dieser **1. Aufgabe** ist eine Parabel gegeben. Am Scheitelpunkt der Parabel ist ein Punkt  $A$ , mit dem die Parabel verschoben werden kann.  $A$  kann nur entlang der y-Achse und nicht in andere Richtungen bewegt werden.
- Die Aufgabe ist, den Punkt so zu verschieben, dass die Nullstellen der Parabel bei  $-2$  und  $2$  liegen. Dies ist der Fall, wenn  $A$  die Koordinaten  $(0 \mid -4)$  besitzt, also **2. so**:



## STACK // Aufgabe 3: Fragetitel, Fragetext und Board

- Starten Sie erneut mit einer neuen STACK-Aufgabe und geben Sie ihr einen geeigneten Fragetitel.
- Tragen Sie in das Feld „Fragetext“ zwischen die HTML-`<p>`-Tags wieder den Aufgabentext ein, der den Studierenden angezeigt werden soll, zum Beispiel so:  
`<p>Verschieben Sie den Punkt so, dass die Nullstellen der Parabel bei  $(x_1 = -2)$  und  $(x_2 = 2)$  liegen.</p>`
- Fügen Sie nun wieder die Platzhalter für den JSXGraph-Block ein. Die Variable für das Board können Sie aus der vorherigen Aufgabe übernehmen:

```
[[jsxgraph]]
const board = JXG.JSXGraph.initBoard(divid, {
  boundingbox: [-10, 5, 10, -5],
  axis: true,
  showCopyright: false
});
[/jsxgraph]]
```

## STACK // Aufgabe 3: Glider hinzufügen

- In der Grafik soll ein Punkt erscheinen, der nur auf der y-Achse bewegt werden kann. Solch ein Punkt, der an ein Objekt gebunden ist, heißt in JSXGraph „Glider“.
- Erstellen Sie zunächst eine Variable `linie` für eine Linie, die auf der y-Achse liegt (Objektyp `line`, vgl. Folie [13](#)):

```
const linie = board.create("line", [ [0, -1], [0, 1] ], {  
  visible: false  
});
```

Bewirkt, dass die Linie  
unsichtbar ist

Koordinaten der Punkte, die  
die Linie definieren

- Erstellen Sie nun noch eine Variable `g` für den Glider an sich, also den Punkt, der nur auf `linie` bewegt werden kann (Objektyp `glider`):

```
const g = board.create("glider", [ 0, 0, linie ]);
```

Startkoordinaten,  
die der Glider  
haben soll

Name der  
Variable für  
die Linie

## STACK // Aufgabe 3: Funktionsgraph hinzufügen

- Fügen Sie dem Board als nächstes den Funktionsgraph für die Parabel hinzu (Objektyp `functiongraph`, vgl. Folie [21](#)).
- Die Funktion, die hier geplottet werden soll, hat die Funktionsgleichung  $f(x) = x^2 + y_g$ , wobei  $y_g$  die y-Koordinate des Gliders ist:

```
const f = board.create("functiongraph", [  
  x=> x*x + g.Y()  
]);
```

$x^2 = x \cdot x$

y-Koordinate des Gliders

- Für die Verbindung des Gliders mit dem Eingabefeld beachten Sie Folie [14](#). Analog zu Punkten (Objektyp `point`) kann bei Glidern ebenfalls die Funktion `bind_point` verwendet werden.
- Blenden Sie dann wieder das Eingabefeld aus (vgl. Folie [15](#)).

*Hinweis: Allgemein können Potenzen  $a^n$  in JavaScript als `Math.pow(a, n)` angegeben werden.*

## STACK // Aufgabe 3: Fertigstellung der Aufgabe

- Erstellen Sie in den Aufgabenvariablen eine Variable für die korrekte Antwort (z. B. `ta1: [0, -4];`), verwenden Sie diese Variable im Feld „Musterlösung“ im Bereich „Eingabe“ und achten Sie wieder darauf, dass die Verwendung von Fließkommazahlen erlaubt ist.
- Konfigurieren Sie Ihren Rückmeldebaum wie im **1 Screenshot** zu sehen. Da die x-Koordinate des Gliders immer 0 ist, genügt es, die y-Koordinate `ans1[2]` zu überprüfen.

The screenshot shows the configuration for 'Knoten 1' in the STACK system. It includes a description field, an answer verification dropdown set to 'NumAbsolute', and input fields for 'SAns' (containing 'ans1[2]'), 'TAns' (containing '-4'), and 'Test-Optionen' (containing '0.1'). There is also a 'Feedback unterdrücken' dropdown set to 'Nein'.

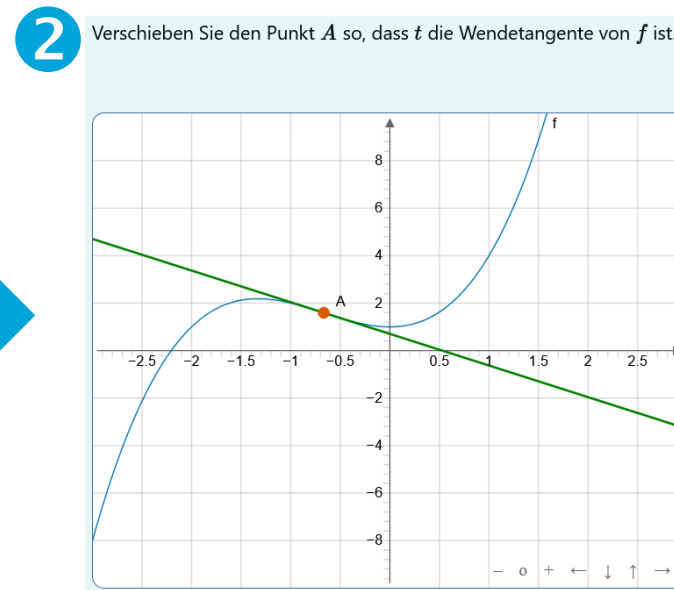
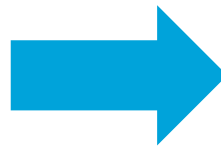
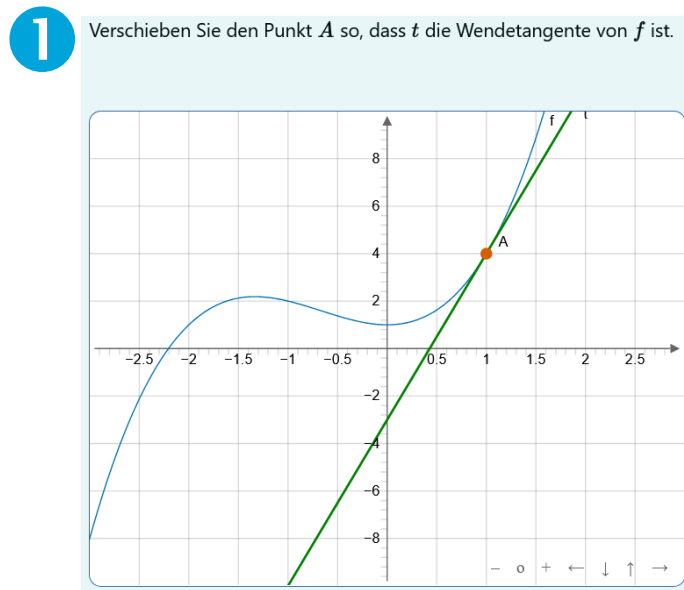
| Knoten 1                        |                       |
|---------------------------------|-----------------------|
| Beschreibung                    |                       |
| Antwortüberprüfung: NumAbsolute |                       |
| SAns                            | TAns                  |
| ans1[2]                         | -4                    |
| Test-Optionen                   | Feedback unterdrücken |
| 0.1                             | Nein                  |

# Aufgabe 4: Wendetangente markieren

*Stand: 02.10.2026*

# STACK // Aufgabe 4: Wendetangente markieren

- In dieser **1. Aufgabe** ist der Graph einer Funktion  $f$  gegeben. Auf dem Graphen lässt sich ein Punkt  $A$  (Glider) bewegen. Egal, wohin  $A$  bewegt wird, ist auch immer die Tangente  $t$  an  $f$  in  $A$  zu sehen.
- Die Aufgabe ist den Punkt so zu setzen, dass die Tangente die Wendetangente des Funktionsgraphen ist, also **2. so**:



# STACK // Aufgabe 4: Fragetitel, Aufgabenvariablen, Fragetext und Board

- Starten Sie erneut mit einer neuen STACK-Aufgabe und geben Sie ihr einen geeigneten Fragetitel.
- Mit dieser Aufgabe wird demonstriert, wie Sie Funktionen, die Sie in den STACK-Aufgabenvariablen definiert haben, mit JSXGraph zeichnen können.
- Erstellen Sie dazu eine Aufgabenvariable für die Funktion, zum Beispiel  $fx: x^3+2*x^2+1$ ;
- Erstellen Sie dann eine Variable  $ta1$  für die korrekte Antwort, also den Wendepunkt der Funktion. Dieser liegt bei  $x = -\frac{2}{3}$ . Die y-Koordinate kann Maxima berechnen, indem es  $-\frac{2}{3}$  in den Funktionsterm  $fx$  einsetzt (Funktion `subst`). Es ist sinnvoll, beide Koordinaten in Dezimalzahlen umzuwandeln (Funktion `float`):  
 $ta1: [float(-2/3), float(subst(x=-2/3, fx))];$
- Tragen Sie in das Feld „Fragetext“ zwischen die HTML-`<p>`-Tags wieder den Aufgabentext ein, der den Studierenden angezeigt werden soll, zum Beispiel so:  
`<p>Verschieben Sie den Punkt  $\backslash(A\backslash)$  so, dass  $\backslash(t\backslash)$  die Wendetangente von  $\backslash(f\backslash)$  ist.</p>`
- Fügen Sie nun wieder die Platzhalter für den JSXGraph-Block ein. Die Variable für das Board können Sie erneut aus der vorherigen Aufgabe übernehmen und müssen nur das `boundingbox`-Attribut anpassen:

```
const board = JXG.JSXGraph.initBoard(divid, {  
  boundingbox: [-3,10,3,-10],  
  axis:true,  
  showCopyright:false  
});
```

## STACK // Aufgabe 4: Plot der Funktion aus den Aufgabenvariablen

- Um die Funktion aus den STACK-Aufgabenvariablen mit JSXGraph zu zeichnen, wird die folgende Zeile benötigt. Hier wird die Maxima-Funktion `fx` in eine JavaScript-Funktion übersetzt und als JavaScript-Variable `f` gespeichert:

```
const f = board.jc.snippet("#{fx#}", true, "x", true);
```

Maxima-Variable zwischen  
Platzhaltern (vgl. Folie [21](#))

Variable, von der die  
Funktion abhängt

- Mit einer zweiten Variable `graph` muss nun noch der Graph der oben definierten JavaScript-Funktion gezeichnet werden (Objektyp `functiongraph`):

```
const graph = board.create("functiongraph", [f], {  
  name: "f",  
  withLabel: true  
});
```

Name, der neben  
dem Graph angezeigt  
werden soll

Bewirkt, dass der Name `f`  
tatsächlich angezeigt wird

Name der JavaScript-Variable  
für die Funktion

## STACK // Aufgabe 4: Glider und Tangente hinzufügen

- Fügen Sie dem Board nun einen Glider `g` hinzu (Objektyp `glider`), der sich auf dem Graphen `graph` bewegen kann:

```
const g = board.create("glider", [1, 4, graph]);
```

Startkoordinaten des Gliders

Variable für den Funktionsgraph

- Die Tangente an `graph` in `g` kann JSXGraph automatisch konstruieren (Objektyp `tangent`):

Name, der neben dem Graph angezeigt werden soll

Bewirkt, dass der Name `f` tatsächlich angezeigt wird

```
const t = board.create("tangent", [g], {  
  name: "t",  
  withLabel: true,  
  strokeColor: "green"  
});
```

Bewirkt, dass die Tangente eine grüne Linie besitzt

Name der JavaScript-Variable für die Funktion

## STACK // Aufgabe 4: Fertigstellung der Aufgabe

- Für die Verbindung des Gliders mit dem Eingabefeld beachten Sie Folie [14](#). Analog zu Punkten (Objektyp `point`) kann bei Glidern ebenfalls die Funktion `bind_point` verwendet werden.
- Blenden Sie dann wieder das Eingabefeld aus (vgl. Folie [15](#)).
- Verwenden Sie die bereits erstellte Aufgabenvariable `ta1` im Feld „Musterlösung“ im Bereich „Eingabe“ und achten Sie wieder darauf, dass die Verwendung von Fließkommazahlen erlaubt ist.
- Konfigurieren Sie Ihren Rückmeldebaum wie im **1 Screenshot** zu sehen. Da die y-Koordinate des Gliders ohnehin dem Wert der Funktion an der Stelle der x-Koordinate entspricht, genügt es, die x-Koordinate `ans1[1]` zu überprüfen.

**Knoten 1**

?

Beschreibung

Antwortüberprüfung

NumAbsolute

SAns

ans1[1]

TAns

ta1[1]

Test-Optionen

0.1

Feedback unterdrücken

Nein

## STACK // Abschließende Hinweise

- Sie können problemlos auch mehrere JSXGraph-Objekte gleichzeitig an STACK-Eingabefelder binden. Erstellen Sie dazu ein Eingabefeld pro JSXGraph-Objekt. Beachten Sie, dass Sie dann im öffnenden `[[jsxgraph]]`-Tag mehrere Bindungen konfigurieren müssen. Beispiel:

```
[[jsxgraph input-ref-ans1="ans1Ref" input-ref-ans2="ans2Ref"]]  
...  
  
const p = board.create("point", [1,1]);  
stack_jxg.bind_point(ans1Ref,p);  
  
const s = board.create("slider", [[2,-4],[8,-4],[0,1,5]]);  
stack_jxg.bind_slider(ans2Ref,s);  
[[/jsxgraph]]
```

- Es ist möglich, den JavaScript-Code für JSXGraph-Grafiken in einem externen Texteditor (z. B. „Visual Studio Code“ von Microsoft) vorzuschreiben. Das hat den Vorteil, dass Sie von Features wie Code-Highlighting und Auto-Vervollständigung profitieren können, die der Moodle-Texteditor aktuell nicht bietet. Es ist auch möglich, eine im Texteditor erstellte JSXGraph-Grafik im Browser als Vorschau anzusehen. Eine Vorlage für eine entsprechende HTML-Datei finden Sie in unserem Moodle-Kurs.

# STACK // Moodle-Kurs „Tutorials STACK“

- Dort finden Sie Videotutorials, Beispielaufgaben, weitere Klickanleitung und mehr
- Link: <https://moodle.ruhr-uni-bochum.de/course/view.php?id=22225>
- Passend zu dieser Klickanleitung empfehlen wir Ihnen:
  - Videotutorial „Interaktive Grafiken mit JSXGraph“
  - Dokument mit einer Liste von JSXGraph-Objekttypen



The screenshot shows the Moodle course page for 'Tutorials zu STACK' at the Ruhr-Universität Bochum (RUB). The header includes the RUB logo and navigation links: Dashboard, LEARN2Gether, OpenRUB, and Support. The course title 'Tutorials zu STACK' is prominently displayed next to a colorful staircase graphic. Below the title, a paragraph explains that STACK is a Moodle question type for digital mathematical tasks. Another paragraph mentions that the course covers various aspects of STACK through video tutorials and example tasks. A final paragraph states that the course is open and accessible without login, but an account is needed for full functionality.

**RUB** Dashboard LEARN2Gether OpenRUB Support Teilnehmer/in

Alles einklappen

## Tutorials zu STACK

STACK ist ein Moodle-Fragetyp, mit dem sich geschlossene und offene digitale mathematische Aufgaben umsetzen lassen, die automatisch ausgewertet werden. Dabei ist es möglich, dass STACK Aufgabenstellungen zufällig generiert und den Lernenden ein differenziertes und adaptives Feedback bereitstellt.

In den Abschnitten des Kurses werden unterschiedliche Aspekte des Fragetyps STACK in Form von Videotutorials und Beispielaufgaben vermittelt. Zum Start empfehlen wir Ihnen die Inhalte des Abschnitts "Einführung in STACK" und "Eine STACK-Aufgabe erstellen", die Grundlagen vermitteln. Die weiteren Abschnitte können in beliebiger Reihenfolge bearbeitet werden.

Dieser Kurs ist komplett öffentlich und ohne Login zugänglich. Um alle Funktionen uneingeschränkt nutzen zu können, ist allerdings ein Account für das Moodle der Ruhr-Universität Bochum notwendig. Falls Sie noch keinen Account besitzen, können Sie sich über diesen Link kostenlos und unkompliziert einen Account erstellen. Danach müssen Sie sich noch in den Kurs einschreiben. Dies funktioniert unter diesem Link.



Dieses Dokument steht – ausgenommen verwendeter Logos bzw. Wort-/Bildmarken, Screenshots und anders gekennzeichnete Elemente – unter einer CC BY-SA 4.0 Lizenz. Sie dürfen die Inhalte **vervielfältigen, verwenden, verarbeiten, vermischen und verbreiten** unter der **Bedingung**, dass Sie die Originalautor:innen nennen und Beiträge, die durch Remix, Veränderung oder Aufbau auf diesem Video entstehen, nur unter derselben Lizenz wie das Original verbreiten.



#### *Anmerkung für Nachnutzende | Version 1.1.a vom 02.10.2026*

##### **Empfohlene Zitation:**

„Parametrisierbare Aufgaben mit STACK: Interaktive Grafiken mit JSXGraph“, Team E-Learning@Hochschule Ruhr West von Jonas Lache, ist lizenziert unter einer CC BY-SA 4.0 International Lizenz (<https://creativecommons.org/licenses/by-sa/4.0/>).

Ausgenommen von der Lizenz sind die verwendeten Wort-/Bildmarken und alle anders gekennzeichneten Elemente.

Dieses Video basiert auf dem Video „Interaktive Grafiken mit JSXGraph“, Team E-Learning@Hochschule Ruhr West von Jonas Lache, das unter CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0/>) im Moodle-Kurs „E-Learning@HRW – Selbstlernkurs STACK“ (<https://elearning.hs-ruhrwest.de/course/view.php?id=24999>) veröffentlicht wurde.

#### *Zusätzliche Informationen für Nachnutzende:*

##### **Ursprungsort:**

Moodle-Kurs „E-Learning@HRW – Selbstlernkurs STACK“ (<https://elearning.hs-ruhrwest.de/course/view.php?id=24999>), Stand 02.10.2026, Version 1.1.a

##### **Urheber:in(nen):**

Jonas Lache, Team E-Learning@Hochschule Ruhr West, Mail: [elearning@hs-ruhrwest.de](mailto:elearning@hs-ruhrwest.de)

##### **Bildnachweise (einzeln):**

– Logo des Assessment-Systems STACK: [stack-assessment.org](https://stack-assessment.org), CC BY-SA 4.0