# ChatGPT prompty

is correct? y = a \* (1 / Math.tan(b \* x)) + c;

replace in array every x with y in js.

What are the ways to make the svg zoomable?

Do you think caching results and reading them from the memory will be faster than calculating this function again?  
let evaluator = (expression: string[], x: number) => { expression = expression.map((item) => item === "x" ? x.toString() : item.toString() ); for (const { symbol, fn } of brackets) { for (let i = 0; i < expression.length; i++) { if (expression[i] === symbol[0]) { let openIndex = i; let depth = 1; if (symbol[0] === "|") { // Special handling for absolute value bars let openIndex = i; for (let j = i + 1; j < expression.length; j++) { if (expression[j] === "|") { const subExpr = expression.slice(openIndex + 1, j); let result = evaluator(subExpr, x); result = Math.abs(parseFloat(result)).toString(); expression.splice(openIndex, j - openIndex + 1, result); i = openIndex - 1; break; } } } for (let j = i + 1; j < expression.length; j++) { if (expression[j] === symbol[0]) depth++; if (expression[j] === symbol[1]) depth--; if (depth === 0) { const subExpr = expression.slice(openIndex + 1, j); let result = evaluator(subExpr, x); expression.splice(openIndex, j - openIndex + 1, result); // replace ( ... ) with result i = openIndex - 1; // rewind to recheck break; } } } } } for (const { symbol, fn } of functions) { for (let i = expression.length - 1; i >= 0; i--) { if (expression[i] === symbol) { const right = parseFloat(expression[i + 1]); if (!isNaN(right)) { const result = fn(right); expression[i] = result.toString(); expression.splice(i + 1, 1); // remove operator and right operand } else { throw new Error(Invalid operands for '${symbol}'); } } } } for (const { symbol, fn } of OPERATORS) { for (let i = expression.length - 1; i >= 0; i--) { if (expression[i] === symbol) { const left = parseFloat(expression[i - 1]); const right = parseFloat(expression[i + 1]); if (!isNaN(left) && !isNaN(right)) { const result = fn(left, right); expression[i - 1] = result.toString(); expression.splice(i, 2); // remove operator and right operand } } } } return expression[0]; };

I have a problem with react re-renders. I added zooming in and out to my svg component. But every zoom changes the viewbox, so it recalculates all of it again, I would rather save everything into lets say session storage and on zoom it would only calculate the missing parts.

how does Geogebra work? my app starts throttling when I use zoom because it has to recalculate the data I am going to counter thiss by implementing IndexedDb because then I will be able to calculate only parts of data. but when I open geogebra I don't see any storage, server or indexedDb

what is in-memory data storage? how to use it?

Compare InsexedDb, In-Memory Storage and localStorage

Uncaught TypeError: Cannot read properties of undefined (reading 'push') at General (General.tsx:80:16) let a: coords = { x: x, y: y }; paths[j].push(a);

Write me a useDebounce hook in react tsx.

I decided that when its zooming it would not display entire graph until its done. How to achieve that?

write me a useMemo for expresion string [] and viewBox type viewbox

can we mathematically expect how much is the y in function going to change? for example if we could expect asymptotes thanks to that

could you update it for color?   
import React, { useState } from "react"; interface UserInputProps { onSubmitExpressions: (functions: string[]) => void; } const UserInput: React.FC<UserInputProps> = ({ onSubmitExpressions }) => { const [functions, setFunctions] = useState<string[]>([""]); const maxFunctions = 10; const handleInputChange = (index: number, value: string) => { const updatedFunctions = [...functions]; updatedFunctions[index] = value; setFunctions(updatedFunctions); if ( value.trim() !== "" && index === functions.length - 1 && functions.length < maxFunctions ) { setFunctions([...updatedFunctions, ""]); } }; const handleSubmit = (event: React.FormEvent) => { event.preventDefault(); const validFunctions = functions.filter((fn) => fn.trim() !== ""); onSubmitExpressions(validFunctions); }; return ( <form onSubmit={handleSubmit}> {functions.map((fn, index) => ( <div key={index}> <label> Function {index + 1}: <input type="text" value={fn} onChange={(e) => handleInputChange(index, e.target.value)} placeholder="Enter a mathematical function" /> <input type="color" value="#000000" // Default color value onChange={(e) => console.log(Color selected for function ${index + 1}:, e.target.value)} /> </label> </div> ))} {functions.length === maxFunctions && ( <p>You have reached the maximum number of functions.</p> )} <button type="submit">Submit</button> </form> ); }; export default UserInput;

could you add that the evaluator would also account the constants array?   
export const evaluator = (expression: string[], x: number) => { expression = expression.map((item) => item === "x" ? x.toString() : item.toString() ); for (const { symbol, fn } of brackets) { for (let i = 0; i < expression.length; i++) { if (expression[i] === symbol[0]) { let openIndex = i; let depth = 1; if (symbol[0] === "|") { let openIndex = i; for (let j = i + 1; j < expression.length; j++) { if (expression[j] === "|") { const subExpr = expression.slice(openIndex + 1, j); let result = evaluator(subExpr, x); result = Math.abs(parseFloat(result)).toString(); expression.splice(openIndex, j - openIndex + 1, result); i = openIndex - 1; break; } } } for (let j = i + 1; j < expression.length; j++) { if (expression[j] === symbol[0]) depth++; if (expression[j] === symbol[1]) depth--; if (depth === 0) { const subExpr = expression.slice(openIndex + 1, j); let result = evaluator(subExpr, x); expression.splice(openIndex, j - openIndex + 1, result); // replace ( ... ) with result i = openIndex - 1; // rewind to recheck break; } } } } } for (const { symbol, fn } of functions) { for (let i = expression.length - 1; i >= 0; i--) { if (expression[i] === symbol) { const right = parseFloat(expression[i + 1]); if (!isNaN(right)) { const result = fn(right); expression[i] = result.toString(); expression.splice(i + 1, 1); // remove operator and right operand } else { throw new Error(Invalid operands for '${symbol}'); } } } } for (const { symbol, fn } of OPERATORS) { for (let i = expression.length - 1; i >= 0; i--) { if (expression[i] === symbol) { const left = parseFloat(expression[i - 1]); const right = parseFloat(expression[i + 1]); if (!isNaN(left) && !isNaN(right)) { const result = fn(left, right); expression[i - 1] = result.toString(); expression.splice(i, 2); // remove operator and right operand } } } } return expression[0]; }; const OPERATORS = [ { symbol: "^", fn: (a: number, b: number) => Math.pow(a, b) }, { symbol: "\*", fn: (a: number, b: number) => a \* b }, { symbol: "/", fn: (a: number, b: number) => a / b }, { symbol: "+", fn: (a: number, b: number) => a + b }, { symbol: "-", fn: (a: number, b: number) => a - b }, ]; const functions = [ { symbol: "sin", fn: (a: number) => Math.sin(a) }, { symbol: "cos", fn: (a: number) => Math.cos(a) }, { symbol: "tan", fn: (a: number) => Math.tan(a) }, { symbol: "log", fn: (a: number) => Math.log10(a) }, { symbol: "cotan", fn: (a: number) => 1/Math.tan(a) }, { symbol: "sqrt", fn: (a: number) => Math.sqrt(a) }, ]; const constants = [ { symbol: "pi", value: Math.PI }, { symbol: "e", value: Math.E }, { symbol: "phi", value: (1 + Math.sqrt(5)) / 2 }, { symbol: "tau", value: 2 \* Math.PI }, { symbol: "gamma", value: 0.5772156649 }, { symbol: "catalan", value: 0.9159655941 }, { symbol: "apery", value: 1.2020569032 }, { symbol: "eulerMascheroni", value: 0.5772156649 }, { symbol: "goldenRatioConjugate", value: (Math.sqrt(5) - 1) / 2 }, ]; const brackets = [ { symbol: ["(", ")"], fn: (a: number) => a }, { symbol: ["|", "|"], fn: (a: number) => Math.abs(a) }, ]; const variables = [ { symbol: "x", value: 0 }, { symbol: "y", value: 0 }, { symbol: "z", value: 0 }, ];

how to detect asymptote?

import React, { useEffect, useRef, useState } from "react"; import General from "./General"; import { parseExpression } from "./utils/ParseExpression" import generateGrid from "./utils/generateGrid"; import { ViewBox } from "./types"; import useDebounce from "./CustomHooks/useDebounce"; // Define the ViewBox type import { getFunctionDataByExpression, replaceFunction, addFunction, flushFunctionData } from './SessionStorage'; import { FunctionData, reqs } from './types'; let LibraryController = ({ reqs, params, }: { reqs: reqs[]; params: { x: number; y: number; width: number; height: number }; }) => { const [viewBox, setViewBox] = useState({ x: params.x, y: params.y, width: params.width, height: params.height, }); const [debouncedViewBox, setDebouncedViewBox] = useState(viewBox); useDebounce(() => { setDebouncedViewBox(viewBox); }, 300, [viewBox]); let data: React.ReactElement[] = []; for (let i = 0; i < reqs.length; i++) { let expression = parseExpression(reqs[i].expression); let storedExpression: FunctionData | undefined = getFunctionDataByExpression(expression); if (storedExpression) { data[i] = <General expression={expression} viewBox={debouncedViewBox} storedExpression={storedExpression} />; }else { data[i] =<General expression={expression} viewBox={debouncedViewBox} />; } } ... LibraryController.tsx:29 Uncaught TypeError: Cannot read properties of undefined (reading 'length') at LibraryController (LibraryController.tsx:29:35)

would you cite according to ISO 690 this web page? <https://www.w3schools.com/graphics/svg_path.asp>

and this one as well? <https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorials/SVG_from_scratch/Paths>

could you do this pdf as well? But I wonder as what should I put it. Document from web? <http://www.jreichl.com/matematika/vyuka/texty/fce_prehled.pdf>

mohl jsi to zretušovat a dopsat?   
Druhý způsob který jsem měl na výběr využívá beziérovi křivky. Tyto křivky se zapisují: d="M 100 350 Q 225 50 350 350" Na citaci výše si můžeme všimnout písmene Q to značí quadratic-bezíer curve. Po tomto písmenu následují 2 souřadnice. Ta druhá [350, 350] je konečná, kde ta cesta buke končit. A ta první je „control-point“ Křivky fungují tím způsobem, že…

I would like to write my very own evaluator, how does it work?

whats wrong there? for (let i = 0; i < expression.length; i++) { if (expression[i] === "^") { let a = Math.pow( parseFloat(expression[i - 1]),parseFloat(expression[i + 1])); expression[i - 1] = a.toString(); expression.splice(i, 2); } }

could you do same for this?  
 for (let i = 0; i < expression.length; i++) { if (expression[i] === "\*") { let a = Math.pow( parseFloat(expression[i - 1]),parseFloat(expression[i + 1])); expression[i - 1] = a.toString(); expression.splice(i, 2); } } for (let i = 0; i < expression.length; i++) { if (expression[i] === "/") { let a = Math.pow( parseFloat(expression[i - 1]),parseFloat(expression[i + 1])); expression[i - 1] = a.toString(); expression.splice(i, 2); } } for (let i = 0; i < expression.length; i++) { if (expression[i] === "+") { let a = Math.pow( parseFloat(expression[i - 1]),parseFloat(expression[i + 1])); expression[i - 1] = a.toString(); expression.splice(i, 2); } } for (let i = 0; i < expression.length; i++) { if (expression[i] === "-") { let a = Math.pow( parseFloat(expression[i - 1]),parseFloat(expression[i + 1])); expression[i - 1] = a.toString(); expression.splice(i, 2); } }

this looks good, but I would like to add brackets rn. I have a solution that if it would detect an opening bracket it would create the same function recursively.

I got this svg and parts are not showing. M -15 -1.9285714285714286 L -14.8 -1.9275362318840579 L -14.600000000000001 -1.9264705882352942 L -14.400000000000002 -1.9253731343283582 L -14.200000000000003 -1.9242424242424243 L -14.000000000000004 -1.9230769230769231 L -13.800000000000004 -1.921875 L -13.600000000000005 -1.9206349206349207 L -13.400000000000006 -1.9193548387096775 L -13.200000000000006 -1.918032786885246 L -13.000000000000007 -1.9166666666666667 L -12.800000000000008 -1.9152542372881356 L -12.600000000000009 -1.913793103448276 L -12.40000000000001 -1.9122807017543861 L -12.20000000000001 -1.9107142857142858 L -12.00000000000001 -1.9090909090909092 L -11.800000000000011 -1.9074074074074074 L -11.600000000000012 -1.9056603773584906 L -11.400000000000013 -1.903846153846154 L -11.200000000000014 -1.9019607843137256 L -11.000000000000014 -1.9000000000000001 L -10.800000000000015 -1.8979591836734695 L -10.600000000000016 -1.8958333333333335 L -10.400000000000016 -1.893617021276596 L -10.200000000000017 -1.8913043478260871 L -10.000000000000018 -1.888888888888889 L -9.800000000000018 -1.8863636363636367 L -9.60000000000002 -1.8837209302325584 L -9.40000000000002 -1.8809523809523812 L -9.20000000000002 -1.8780487804878052 L -9.000000000000021 -1.8750000000000002 L -8.800000000000022 -1.8717948717948723 L -8.600000000000023 -1.8684210526315794 L -8.400000000000023 -1.8648648648648654 L -8.200000000000024 -1.8611111111111116 L -8.000000000000025 -1.8571428571428577 L -7.800000000000025 -1.8529411764705888 L -7.6000000000000245 -1.848484848484849 L -7.400000000000024 -1.8437500000000007 L -7.200000000000024 -1.8387096774193554 L -7.000000000000024 -1.833333333333334 L -6.800000000000024 -1.8275862068965525 L -6.600000000000024 -1.8214285714285723 L -6.4000000000000234 -1.8148148148148155 L -6.200000000000023 -1.8076923076923086 L -6.000000000000023 -1.800000000000001 L -5.800000000000023 -1.7916666666666676 L -5.600000000000023 -1.782608695652175 L -5.400000000000023 -1.7727272727272738 L -5.200000000000022 -1.7619047619047632 L -5.000000000000022 -1.7500000000000013 L -4.800000000000022 -1.7368421052631595 L -4.600000000000022 -1.7222222222222239 L -4.400000000000022 -1.7058823529411784 L -4.2000000000000215 -1.687500000000002 L -4.000000000000021 -1.666666666666669 L -3.800000000000021 -1.6428571428571455 L -3.600000000000021 -1.6153846153846185 L -3.400000000000021 -1.583333333333337 L -3.2000000000000206 -1.5454545454545496 L -3.0000000000000204 -1.500000000000005 L -2.8000000000000203 -1.4444444444444506 L -2.60000000000002 -1.3750000000000078 L -2.40000000000002 -1.2857142857142958 L -2.2000000000000197 -1.1666666666666803 L -2.0000000000000195 -1.0000000000000195 L -1.8000000000000196 -0.7500000000000306 L -1.6000000000000196 -0.3333333333333879 L -1.4000000000000197 0.49999999999987704 L -1.2000000000000197 2.999999999999507 L -1.0000000000000198 50602243004160.875 L -0.8000000000000198 -7.000000000000496 L -0.6000000000000199 -4.500000000000124 L -0.40000000000001984 -3.666666666666722 L -0.20000000000001983 -3.250000000000031 L -1.9817480989559044e-14 -3.00000000000002 L 0.1999999999999802 -2.8333333333333472 L 0.3999999999999802 -2.7142857142857246 L 0.5999999999999802 -2.625000000000008 L 0.7999999999999803 -2.5555555555555616 L 0.9999999999999802 -2.5000000000000053 L 1.1999999999999802 -2.4545454545454586 L 1.3999999999999801 -2.41666666666667 L 1.59999999999998 -2.3846153846153877 L 1.79999999999998 -2.35714285714286 L 1.99999999999998 -2.3333333333333357 L 2.19999999999998 -2.3125000000000018 L 2.3999999999999804 -2.294117647058825 L 2.5999999999999805 -2.277777777777779 L 2.7999999999999807 -2.263157894736844 L 2.999999999999981 -2.250000000000001 L 3.199999999999981 -2.238095238095239 L 3.3999999999999813 -2.2272727272727284 L 3.5999999999999814 -2.217391304347827 L 3.7999999999999816 -2.2083333333333344 L 3.999999999999982 -2.2000000000000006 L 4.1999999999999815 -2.192307692307693 L 4.399999999999982 -2.185185185185186 L 4.599999999999982 -2.1785714285714293 L 4.799999999999982 -2.172413793103449 L 4.999999999999982 -2.166666666666667 L 5.199999999999982 -2.1612903225806455 L 5.399999999999983 -2.1562500000000004 L 5.599999999999983 -2.151515151515152 L 5.799999999999983 -2.147058823529412 L 5.999999999999983 -2.1428571428571432 L 6.199999999999983 -2.1388888888888893 L 6.3999999999999835 -2.1351351351351355 L 6.599999999999984 -2.1315789473684212 L 6.799999999999984 -2.128205128205128 L 6.999999999999984 -2.1250000000000004 L 7.199999999999984 -2.1219512195121952 L 7.399999999999984 -2.119047619047619 L 7.5999999999999845 -2.116279069767442 L 7.799999999999985 -2.1136363636363638 L 7.999999999999985 -2.111111111111111 L 8.199999999999985 -2.108695652173913 L 8.399999999999984 -2.1063829787234045 L 8.599999999999984 -2.104166666666667 L 8.799999999999983 -2.102040816326531 L 8.999999999999982 -2.1 L 9.199999999999982 -2.0980392156862746 L 9.39999999999998 -2.0961538461538463 L 9.59999999999998 -2.0943396226415096 L 9.79999999999998 -2.0925925925925926 L 9.999999999999979 -2.0909090909090913 L 10.199999999999978 -2.0892857142857144 L 10.399999999999977 -2.0877192982456143 L 10.599999999999977 -2.086206896551724 L 10.799999999999976 -2.0847457627118646 L 10.999999999999975 -2.0833333333333335 L 11.199999999999974 -2.0819672131147544 L 11.399999999999974 -2.080645161290323 L 11.599999999999973 -2.0793650793650795 L 11.799999999999972 -2.078125 L 11.999999999999972 -2.076923076923077 L 12.19999999999997 -2.075757575757576 L 12.39999999999997 -2.074626865671642 L 12.59999999999997 -2.073529411764706 L 12.799999999999969 -2.0724637681159424 L 12.999999999999968 -2.0714285714285716 L 13.199999999999967 -2.070422535211268 L 13.399999999999967 -2.0694444444444446 L 13.599999999999966 -2.068493150684932 L 13.799999999999965 -2.0675675675675675 L 13.999999999999964 -2.066666666666667 L 14.199999999999964 -2.0657894736842106 L 14.399999999999963 -2.064935064935065 L 14.599999999999962 -2.0641025641025643 L 14.799999999999962 -2.0632911392405067 L 14.999999999999961

could you cite this according to ČSN ISO 690 norm? <https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API>

and this as well? <https://react.dev/reference/react/useMemo>

Could you make me a function that would take as parameter 2 coordinates and it would return delta y? in ts

Uncaught TypeError: Cannot read properties of undefined (reading '0') at computePartialGraph (General.tsx:113:47) console.log(storedPaths[storedsegmentIndex][0].x, i)

General.tsx:120 Uncaught TypeError: Cannot read properties of undefined (reading 'push') at computePartialGraph (General.tsx:120:32) const computePartialGraph = ( expression: string[], viewBox: ViewBox, storedExpression: FunctionData, color?: string ) => { let localStep = viewBox.width / 10000; let localLastY = 1000; let localPaths: coords[][] = []; let localPathArray: string[] = []; let localLock = true; let segmentIndex = 0; let threshold = viewBox.width / 10000 \* 30; const storedPaths = storedExpression.pathArray; let recentPoints: coords[] = []; let storedsegmentIndex = 0; for (let i = viewBox.x; i < viewBox.x + viewBox.width + localStep; i += localStep) { if ( storedPaths[storedsegmentIndex] && storedPaths[storedsegmentIndex][0].x < i) { console.log("fire ", storedsegmentIndex, storedPaths[storedsegmentIndex][0].x, i) i = storedPaths[storedsegmentIndex][storedPaths[storedsegmentIndex].length - 1].x; error>>localPaths[segmentIndex].push(...storedPaths[storedsegmentIndex]); storedsegmentIndex++; } let x = i; let y = parseFloat(evaluator(expression, i)); if (isNaN(y)) {localLock = true; continue;} /\* let ans = deltaY({x: x, y: y}, {x: x, y: localLastY}) if ( curvatureScore(recentPoints) < 0.00001 && ans < 0.1) { console.log("curvature fire ", curvatureScore(recentPoints)) localStep = viewBox.width / 100; } if ( ans > threshold) { console.log("threshhold overwrite ") localStep = viewBox.width / 100000; } recentPoints.push({ x, y }); if (recentPoints.length > 5) { recentPoints.shift(); // remove the oldest point } \*/ if (y > -viewBox.y + 5) continue if (y < viewBox.y - 5) continue; if (Math.abs(localLastY) > -viewBox.y && Math.abs(y) > -viewBox.y && localLastY \* y < 0) { localStep = viewBox.width / 10000; localLastY = viewBox.height; segmentIndex++; localLock = true; } localLastY = y; if (localLock) { const point: coords = { x, y }; if (!localPaths[segmentIndex]) localPaths[segmentIndex] = []; localPaths[segmentIndex][0] = point; localLock = false; } else { localPaths[segmentIndex].push({ x, y }); } } const updatedExpression: FunctionData = { ...storedExpression, pathArray: localPaths, }; const mergedPaths = pathsToDStrings(updatedExpression.pathArray); return mergedPaths.map((d, index) => ( <path key={index} d={d} stroke={color || "black"} fill="none" /> )); };

make this handle if function doesn't exist as well. export const getFunctionDataByExpression = (expression: string[]): FunctionData | undefined => { return appData.functions.find(func => func.expression.some(expr => expression.includes(expr))); };

could you make the zooming and dragging also work for mobile? import React, { useEffect, useRef, useState } from "react"; import General from "./General"; import { parseExpression } from "./utils/ParseExpression" import generateGrid from "./utils/generateGrid"; import { ViewBox } from "./types"; import useDebounce from "./CustomHooks/useDebounce"; // Define the ViewBox type import { FunctionData, reqs } from './types'; import Picker from "./Picker"; let LibraryController = ({ reqs, params, }: { reqs: reqs[]; params: { x: number; y: number; width: number; height: number }; }) => { const [reqsData, setReqsData] = useState<reqs[]>(reqs); useEffect(() => { setReqsData(reqs); }, [reqs]); //data normalization let expressions: FunctionData[] = []; for (let i = 0; i < reqsData.length; i++) { const req = reqsData[i]; if (!req || !req.expression) continue; const expression = parseExpression(req.expression); const color = req.color; expressions[i] = { id: i, color, expression, pathArray: [], }; } //rest const [viewBox, setViewBox] = useState({ x: params.x, y: params.y, width: params.width, height: params.height, }); const [debouncedViewBox, setDebouncedViewBox] = useState(viewBox); useDebounce(() => { setDebouncedViewBox(viewBox); }, 300, [viewBox]); const svgRef = useRef<SVGSVGElement | null>(null); const isPanning = useRef(false); const startPoint = useRef({ x: 0, y: 0 }); const handleMouseDown = (e: React.MouseEvent) => { isPanning.current = true; startPoint.current = { x: e.clientX, y: e.clientY }; }; const handleMouseMove = (e: React.MouseEvent) => { if (!isPanning.current || !svgRef.current) return; const dx = (e.clientX - startPoint.current.x) \* (viewBox.width / svgRef.current.clientWidth); const dy = (e.clientY - startPoint.current.y) \* (viewBox.height / svgRef.current.clientHeight); setViewBox((prev) => { let newX = prev.x - dx; let newY = prev.y - dy; // Clamp the values between -200 and 200 const maxX = 200; const minX = -200; const maxY = 200; const minY = -200; // Prevent panning beyond the range newX = Math.max(minX, Math.min(maxX - prev.width, newX)); newY = Math.max(minY, Math.min(maxY - prev.height, newY)); return { ...prev, x: newX, y: newY, }; }); startPoint.current = { x: e.clientX, y: e.clientY }; }; const endPan = () => { isPanning.current = false; //setMousePos(null); // clear when leaving }; const handleWheel = (e: WheelEvent) => { e.preventDefault(); const zoomFactor = e.deltaY > 0 ? 1.1 : 0.9; setViewBox((prev: { x: number; y: number; width: number; height: number }) => { const centerX = prev.x + prev.width / 2; const centerY = prev.y + prev.height / 2; const newWidth = prev.width \* zoomFactor; const newHeight = prev.height \* zoomFactor; if (newWidth > 400 || newHeight < 0.001) { return prev; // Prevent zooming out too much } return { width: newWidth, height: newHeight, x: centerX - newWidth / 2, y: centerY - newHeight / 2, }; }); }; useEffect(() => {}, [viewBox]); useEffect(() => { const svg = svgRef.current; if (!svg) return; svg.addEventListener("wheel", handleWheel, { passive: false }); return () => { svg.removeEventListener("wheel", handleWheel); }; }, []); const getStrokeWidth = (viewBox: ViewBox): number => { // Adjust multiplier to your liking return viewBox.width / 1000; }; const strokeWidth = getStrokeWidth(viewBox); const grid = generateGrid(viewBox); return ( <svg ref={svgRef} width="1000" height="1000" viewBox={${viewBox.x} ${viewBox.y} ${viewBox.width} ${viewBox.height}} style={{ width: "100%", height: "auto", border: "1px solid black" }} onMouseDown={handleMouseDown} onMouseMove={handleMouseMove} onMouseUp={endPan} onMouseLeave={endPan} > {/\* {mousePos && ( <text x={mousePos.x + 0.5 \* (viewBox.width / 30)} y={mousePos.y - 0.5 \* (viewBox.height / 30)} fontSize={0.5 \* (viewBox.width / 30)} fill="red" alignmentBaseline="middle" style={{ pointerEvents: "none", userSelect: "none" }} > ({mousePos.x.toFixed(2)}, {mousePos.y.toFixed(2)}) </text> )} \*/} <rect x={viewBox.x} y={viewBox.y} width={viewBox.width} height={viewBox.height} fill="white" /> <g fill="none" stroke="black" strokeWidth={strokeWidth}> {viewBox === debouncedViewBox ? <Picker expressions={expressions} params={viewBox} /> : null} </g> <g stroke="lightgray" strokeWidth={strokeWidth}> {grid.vertical.map((x) => ( <line key={v-${x}} x1={x} y1={viewBox.y} x2={x} y2={viewBox.y + viewBox.height} /> ))} {grid.horizontal.map((y) => ( <line key={h-${y}} y1={y} x1={viewBox.x} y2={y} x2={viewBox.x + viewBox.width} /> ))} </g> <line x1={viewBox.x} y1={0} x2={viewBox.x + viewBox.width} y2={0} stroke="black" strokeWidth={strokeWidth}/> <line x1={0} y1={viewBox.y} x2={0} y2={viewBox.y + viewBox.height} stroke="black" strokeWidth={strokeWidth}/> <g fontFamily="Arial" fontSize={0.5 \* (viewBox.width / 30)} fill="black" textAnchor="middle"> {grid.labels.map((label, i) => ( <text key={label-${i}} x={label.x} y={label.y} style={{ pointerEvents: "none", userSelect: "none" }}> {label.text} </text> ))} </g> </svg> ); }; export default LibraryController;

.svg-section { width: 100%; } right now this part overflows to down. I won't wanna use overflow hidden. Its caused because the svg tries to be rectangular can you do it in this class to just fill the screen not craing about rectangularity?

so when user presses button sin I would like to write sin() and put his cursor between brackets

is there a way to draw expression? Like whe user would put x/1 it would show as a fraction. Is there any documentation? How or anything?

okay sso how would you code if you would like to display fraction dynamically for example show me fraction a/b

okay show me the logic behind it

could you insert it into this component? import React, { useState, useRef, useEffect } from "react"; import { reqs } from "./components/types"; import "./App.css"; // Assuming you have some CSS for styling import "./UserInput.css"; // Assuming you have some CSS for styling interface UserInputProps { onSubmitExpressions: (functions: reqs[]) => void; } const UserInput: React.FC<UserInputProps> = ({ onSubmitExpressions }) => { const [functions, setFunctions] = useState<reqs[]>([ { expression: "", color: "#000000" }, ]); const [lastFocusedIndex, setLastFocusedIndex] = useState<number | null>(null); const inputRefs = useRef<(HTMLInputElement | null)[]>([]); const historyRefs = useRef<string[][]>([[""]]); // Each input has its own history const maxFunctions = 10; const maxHistory = 20; const specialButtons = ["sin", "cos", "tan", "^", "√", "π", "e", "(", ")"]; // Set up key listener for Ctrl+Z useEffect(() => { const handleKeyDown = (e: KeyboardEvent) => { if (e.ctrlKey && e.key === "z") { e.preventDefault(); if (lastFocusedIndex === null) return; const history = historyRefs.current[lastFocusedIndex]; if (history && history.length > 1) { history.pop(); // Remove current value const previous = history[history.length - 1]; updateExpression(lastFocusedIndex, previous, false); } } }; window.addEventListener("keydown", handleKeyDown); return () => window.removeEventListener("keydown", handleKeyDown); }, [lastFocusedIndex]); const updateExpression = (index: number, value: string, pushToHistory = true) => { const updated = [...functions]; updated[index].expression = value; setFunctions(updated); if (pushToHistory) { const history = historyRefs.current[index] || []; history.push(value); if (history.length > maxHistory) history.shift(); // Trim old history historyRefs.current[index] = history; } if ( value.trim() !== "" && index === functions.length - 1 && functions.length < maxFunctions ) { setFunctions([...updated, { expression: "", color: "#000000" }]); historyRefs.current.push([""]); } }; const handleExpressionChange = (index: number, value: string) => { updateExpression(index, value); }; const handleColorChange = (index: number, color: string) => { const updated = [...functions]; updated[index].color = color; setFunctions(updated); }; const insertSpecialChar = (char: string) => { if (lastFocusedIndex === null) return; const input = inputRefs.current[lastFocusedIndex]; if (!input) return; if (char.length > 1) { char = char + "()"; // Add opening parenthesis for functions } const start = input.selectionStart ?? 0; const end = input.selectionEnd ?? 0; const value = functions[lastFocusedIndex].expression; let posans = start + char.length; if (char.length > 1) { posans = start + char.length - 1; } const newValue = value.slice(0, start) + char + value.slice(end); updateExpression(lastFocusedIndex, newValue); setTimeout(() => { input.focus(); const pos = posans; input.setSelectionRange(pos, pos); }, 0); }; const handleSubmit = (event: React.FormEvent) => { event.preventDefault(); const validFunctions = functions.filter((fn) => fn.expression.trim() !== ""); onSubmitExpressions(validFunctions); }; useEffect(() => { const firstInput = inputRefs.current[0]; if (firstInput) { firstInput.focus(); setLastFocusedIndex(0); } }, []); return ( <form onSubmit={handleSubmit} className="form-section form-container"> {functions.map((fn, index) => ( <div key={index} className="input-item"> <label className="label-field"> <div className="input-container"> <input className="input-field" type="text" value={fn.expression} onChange={(e) => handleExpressionChange(index, e.target.value)} onFocus={() => setLastFocusedIndex(index)} placeholder="Enter a mathematical function" ref={(el) => { inputRefs.current[index] = el; if (!historyRefs.current[index]) { historyRefs.current[index] = [fn.expression]; } }} /> <input className="input-color-field" type="color" value={fn.color} onChange={(e) => handleColorChange(index, e.target.value)} /> </div> </label> </div> ))} {/\* Shared Special Buttons \*/} <div style={{ margin: "1rem 0", display: "flex", flexWrap: "wrap", gap: "0.5rem" }}> {specialButtons.map((char) => ( <button type="button" key={char} onClick={() => insertSpecialChar(char)} style={{ padding: "4px 8px", fontSize: "0.9rem" }} > {char} </button> ))} </div> {functions.length === maxFunctions && ( <p>You have reached the maximum number of functions.</p> )} <button type="submit">Submit</button> </form> ); }; export default UserInput;

how to design scroll bar?

is there a way to style color field?

could you make this function support () fractions as well? const convertFractions = (expr: string) => { const tokens = expr.split(/([\s+\-\*/()])/).filter((t) => t.trim() !== ""); for (let i = 0; i < tokens.length; i++) { if (tokens[i] === "/") { const left = tokens[i - 1]; const right = tokens[i + 1]; if (left && right) { tokens.splice(i - 1, 3, \\frac{${left}}{${right}}); i--; } } } return tokens.join(""); };

why this works for brackets while not for | as abs value? for (const { symbol, fn } of brackets) { for (let i = 0; i < expression.length; i++) { if (expression[i] === symbol[0]) { let openIndex = i; let depth = 1; let lock = symbol[0] === "|" ? true : false; for (let j = i + 1; j < expression.length; j++) { if (expression[j] === symbol[0]) depth++; if (expression[j] === symbol[1]) depth--; if (depth === 0) { const subExpr = expression.slice(openIndex + 1, j); let result = evaluator(subExpr, x); if (lock) { result = Math.abs(parseFloat(result)).toString(); // Apply absolute value } console.log(result) expression.splice(openIndex, j - openIndex + 1, result); // replace ( ... ) with result i = openIndex - 1; // rewind to recheck break; } } } } } const brackets = [ { symbol: ["(", ")"], fn: (a: number) => a }, { symbol: ["|", "|"], fn: (a: number) => Math.abs(a) }, ]

okay so here I have svg that is made to display a plain function graph from -15 to 15 but as you can see I added dynamical viewbox, so the user can scroll to zoom rn. I would like to make the lines and numbers dynamic as well, because right now it overflows pretty easily. <svg ref={svgRef} width="1000" height="1000" viewBox={${viewBox.x} ${viewBox.y} ${viewBox.width} ${viewBox.height}} style={{ width: "100%", height: "auto", border: "1px solid black" }} > <rect x="-15" y="-15" width="30" height="30" fill="white" /> <g stroke="lightgray" stroke-width="0.1"> <line x1="-15" y1="0" x2="15" y2="0" stroke="black" /> <line y1="-15" y2="15" x1="0" x2="0" stroke="black" /> </g> <g stroke-width="0.1" stroke="lightgray"> <line x1="-10" y1="-15" x2="-10" y2="15" /> <line x1="-5" y1="-15" x2="-5" y2="15" /> <line x1="5" y1="-15" x2="5" y2="15" /> <line x1="10" y1="-15" x2="10" y2="15" /> </g> <g stroke-width="0.1" stroke="lightgray"> <line y1="-10" x1="-15" y2="-10" x2="15" /> <line y1="-5" x1="-15" y2="-5" x2="15" /> <line y1="5" x1="-15" y2="5" x2="15" /> <line y1="10" x1="-15" y2="10" x2="15" /> </g> <g font-family="Arial" font-size="0.5" fill="black" text-anchor="middle"> <text x="-10" y="-0.5"> -10 </text> <text x="-5" y="-0.5"> -5 </text> <text x="5" y="-0.5"> 5 </text> <text x="10" y="-0.5"> 10 </text> <text x="0.5" y="-10"> 10 </text> <text x="0.5" y="-5"> 5 </text> <text x="0.5" y="5"> -5 </text> <text x="0.5" y="10"> -10 </text> </g> <text fontSize="0.5" fill="black"></text> </svg>

okay so could you add that the strokewidth would be dynamic as well?

could you add higher precision here? <g fontFamily="Arial" fontSize={0.5 \* (viewBox.width / 30)} fill="black" textAnchor="middle"> {grid.labels.map((label, i) => ( <text key={label-${i}} x={label.x} y={label.y}> {label.text} </text> ))} </g>

could you add ctrl+z functionality as well?

also could you add that on first mount it would automatically insert into first even when you didn't click into it?

mohl bys nyní krátce popsat tento custom hook a proč je zajímavý? import { useEffect } from "react"; import useTimeout from "./useTimeout"; // adjust path as needed export default function useDebounce( callback: () => void, delay: number, dependencies: React.DependencyList ) { const { reset, clear } = useTimeout(callback, delay); useEffect(reset, [...dependencies, reset]); useEffect(clear, []); } import { useCallback, useEffect, useRef } from "react"; export default function useTimeout(callback: () => void, delay: number) { const callbackRef = useRef(callback); const timeoutRef = useRef<ReturnType<typeof setTimeout> | null>(null); // Always use the latest version of the callback useEffect(() => { callbackRef.current = callback; }, [callback]); const set = useCallback(() => { timeoutRef.current = setTimeout(() => { callbackRef.current(); }, delay); }, [delay]); const clear = useCallback(() => { if (timeoutRef.current !== null) { clearTimeout(timeoutRef.current); } }, []); useEffect(() => { set(); return clear; }, [delay, set, clear]); const reset = useCallback(() => { clear(); set(); }, [clear, set]); return { reset, clear }; }

const handleMouseMove = (e: React.MouseEvent) => { // if (svgRef.current) { // const svgRect = svgRef.current.getBoundingClientRect(); // const svgX = viewBox.x + (e.clientX - svgRect.left) \* (viewBox.width / svgRect.width); // const svgY = viewBox.y + (e.clientY - svgRect.top) \* (viewBox.height / svgRect.height); // setMousePos({ x: svgX, y: svgY }); // } if (!isPanning.current || !svgRef.current) return; const dx = (e.clientX - startPoint.current.x) \* (viewBox.width / svgRef.current.clientWidth); const dy = (e.clientY - startPoint.current.y) \* (viewBox.height / svgRef.current.clientHeight); setViewBox((prev) => ({ ...prev, x: prev.x - dx, y: prev.y - dy, })); startPoint.current = { x: e.clientX, y: e.clientY }; }; could you add there ssome of to make the user not to go out from -200 to 200

I already wrote my useDebounce hook could you implement it there? //toto je napsané přes ai // write me a custome react hook useDebound use ts. import { useEffect } from "react"; import useTimeout from "./useTimeout"; // adjust path as needed export default function useDebounce( callback: () => void, delay: number, dependencies: React.DependencyList ) { const { reset, clear } = useTimeout(callback, delay); useEffect(reset, [...dependencies, reset]); useEffect(clear, []); }

.form-section { max-width: 300px; width: 100%; background-color: #ffffff; margin: 1em; box-shadow: -2px 0 5px rgba(0, 0, 0, 0.1); } problem is that it overflows becausse of padding/margin. how to counter that? it seems like if mergin or padding isn't acounted in max wish

okay now could you add to this component, that it would listen to the position of mause and would display the position in the right bottom corner? import React, { useEffect, useRef, useState, useCallback } from "react"; import General from "./General"; import { parseExpression } from "./utils/ParseExpression" import generateGrid from "./utils/generateGrid"; import { ViewBox } from "./types"; import useDebounce from "./CustomHooks/useDebounce"; import "../App.css"; // Define the ViewBox type import { FunctionData, reqs } from './types'; import Picker from "./Picker"; let LibraryController = ({ reqs, params, }: { reqs: reqs[]; params: { x: number; y: number; width: number; height: number }; }) => { const [reqsData, setReqsData] = useState<reqs[]>(reqs); useEffect(() => { setReqsData(reqs); }, [reqs]); //data normalization let expressions: FunctionData[] = []; for (let i = 0; i < reqsData.length; i++) { const req = reqsData[i]; if (!req || !req.expression) continue; const expression = parseExpression(req.expression); const color = req.color; expressions[i] = { id: i, color, expression, pathArray: [], }; } const updateViewBox = useCallback(() => { const container = document.querySelector(".svg-section"); if (!container) return; const { clientWidth, clientHeight } = container; const aspect = clientWidth / clientHeight; const baseHeight = 4; const newWidth = baseHeight \* aspect; setViewBox({ x: -newWidth / 2, y: -baseHeight / 2, width: newWidth, height: baseHeight, }); }, []); // ✅ Debounced update on resize useDebounce(updateViewBox, 200, [updateViewBox]); // ✅ Immediate update on first mount useEffect(() => { updateViewBox(); // ← call it once right away window.addEventListener("resize", updateViewBox); return () => window.removeEventListener("resize", updateViewBox); }, [updateViewBox]); const lastTouchDistance = useRef<number | null>(null); const isTouchPanning = useRef(false); const lastTouchCenter = useRef<{ x: number; y: number } | null>(null); const getTouchPoint = (touch: Touch, element: SVGSVGElement) => { const rect = element.getBoundingClientRect(); return { x: ((touch.clientX - rect.left) / rect.width) \* viewBox.width + viewBox.x, y: ((touch.clientY - rect.top) / rect.height) \* viewBox.height + viewBox.y, }; }; const getDistance = (touch1: Touch, touch2: Touch) => { return Math.sqrt( Math.pow(touch2.clientX - touch1.clientX, 2) + Math.pow(touch2.clientY - touch1.clientY, 2) ); }; const getCenter = (touch1: Touch, touch2: Touch, element: SVGSVGElement) => { const x = (touch1.clientX + touch2.clientX) / 2; const y = (touch1.clientY + touch2.clientY) / 2; const rect = element.getBoundingClientRect(); return { x: ((x - rect.left) / rect.width) \* viewBox.width + viewBox.x, y: ((y - rect.top) / rect.height) \* viewBox.height + viewBox.y, }; }; const handleTouchStart = (e: TouchEvent) => { if (!svgRef.current) return; if (e.touches.length === 1) { isTouchPanning.current = true; startPoint.current = { x: e.touches[0].clientX, y: e.touches[0].clientY, }; } else if (e.touches.length === 2) { lastTouchDistance.current = getDistance(e.touches[0], e.touches[1]); lastTouchCenter.current = getCenter(e.touches[0], e.touches[1], svgRef.current); } }; const handleTouchMove = (e: TouchEvent) => { if (!svgRef.current) return; if (e.touches.length === 1 && isTouchPanning.current) { const dx = (e.touches[0].clientX - startPoint.current.x) \* (viewBox.width / svgRef.current.clientWidth); const dy = (e.touches[0].clientY - startPoint.current.y) \* (viewBox.height / svgRef.current.clientHeight); setViewBox(prev => ({ ...prev, x: prev.x - dx, y: prev.y - dy, })); startPoint.current = { x: e.touches[0].clientX, y: e.touches[0].clientY, }; } else if (e.touches.length === 2) { const currentDistance = getDistance(e.touches[0], e.touches[1]); const currentCenter = getCenter(e.touches[0], e.touches[1], svgRef.current); if (lastTouchDistance.current && lastTouchCenter.current) { const zoomFactor = lastTouchDistance.current / currentDistance; setViewBox(prev => { const newWidth = prev.width \* zoomFactor; const newHeight = prev.height \* zoomFactor; return { width: newWidth, height: newHeight, x: currentCenter.x - newWidth / 2, y: currentCenter.y - newHeight / 2, }; }); } lastTouchDistance.current = currentDistance; lastTouchCenter.current = currentCenter; } }; const handleTouchEnd = () => { isTouchPanning.current = false; lastTouchDistance.current = null; lastTouchCenter.current = null; }; //rest const [viewBox, setViewBox] = useState({ x: params.x, y: params.y, width: params.width, height: params.height, }); const [debouncedViewBox, setDebouncedViewBox] = useState(viewBox); useDebounce(() => { setDebouncedViewBox(viewBox); }, 300, [viewBox]); const svgRef = useRef<SVGSVGElement | null>(null); const isPanning = useRef(false); const startPoint = useRef({ x: 0, y: 0 }); const handleMouseDown = (e: React.MouseEvent) => { isPanning.current = true; startPoint.current = { x: e.clientX, y: e.clientY }; }; const handleMouseMove = (e: React.MouseEvent) => { if (!isPanning.current || !svgRef.current) return; const dx = (e.clientX - startPoint.current.x) \* (viewBox.width / svgRef.current.clientWidth); const dy = (e.clientY - startPoint.current.y) \* (viewBox.height / svgRef.current.clientHeight); setViewBox((prev) => { let newX = prev.x - dx; let newY = prev.y - dy; // Clamp the values between -200 and 200 const maxX = 200; const minX = -200; const maxY = 200; const minY = -200; // Prevent panning beyond the range newX = Math.max(minX, Math.min(maxX - prev.width, newX)); newY = Math.max(minY, Math.min(maxY - prev.height, newY)); return { ...prev, x: newX, y: newY, }; }); startPoint.current = { x: e.clientX, y: e.clientY }; }; const endPan = () => { isPanning.current = false; //setMousePos(null); // clear when leaving }; const handleWheel = (e: WheelEvent) => { e.preventDefault(); const zoomFactor = e.deltaY > 0 ? 1.1 : 0.9; setViewBox((prev: { x: number; y: number; width: number; height: number }) => { const centerX = prev.x + prev.width / 2; const centerY = prev.y + prev.height / 2; const newWidth = prev.width \* zoomFactor; const newHeight = prev.height \* zoomFactor; if (newWidth > 400 || newHeight < 0.001) { return prev; // Prevent zooming out too much } return { width: newWidth, height: newHeight, x: centerX - newWidth / 2, y: centerY - newHeight / 2, }; }); }; useEffect(() => {}, [viewBox]); useEffect(() => { const svg = svgRef.current; if (!svg) return; svg.addEventListener("wheel", handleWheel, { passive: false }); svg.addEventListener("touchstart", handleTouchStart, { passive: false }); svg.addEventListener("touchmove", handleTouchMove, { passive: false }); svg.addEventListener("touchend", handleTouchEnd); return () => { svg.removeEventListener("wheel", handleWheel); svg.removeEventListener("touchstart", handleTouchStart); svg.removeEventListener("touchmove", handleTouchMove); svg.removeEventListener("touchend", handleTouchEnd); }; }, []); const getStrokeWidth = (viewBox: ViewBox): number => { // Adjust multiplier to your liking return viewBox.width / 1000; }; const strokeWidth = getStrokeWidth(viewBox); const grid = generateGrid(viewBox); return ( <div className="svg-section"> <svg ref={svgRef} className="library-svg" viewBox={${viewBox.x} ${viewBox.y} ${viewBox.width} ${viewBox.height}} style={{ width: "100%", height: "100%", border: "1px solid black" }} onMouseDown={handleMouseDown} onMouseMove={handleMouseMove} onMouseUp={endPan} onMouseLeave={endPan} > <rect x={viewBox.x} y={viewBox.y} width={viewBox.width} height={viewBox.height} fill="white" /> <g fill="none" stroke="black" strokeWidth={strokeWidth}> <g fill="none" stroke="black" strokeWidth={strokeWidth}> {viewBox === debouncedViewBox ? ( <Picker expressions={expressions} params={viewBox} /> ) : null} </g> </g> <g stroke="lightgray" strokeWidth={strokeWidth}> {grid.vertical.map((x) => ( <line key={v-${x}} x1={x} y1={viewBox.y} x2={x} y2={viewBox.y + viewBox.height} /> ))} {grid.horizontal.map((y) => ( <line key={h-${y}} y1={y} x1={viewBox.x} y2={y} x2={viewBox.x + viewBox.width} /> ))} </g> <line x1={viewBox.x} y1={0} x2={viewBox.x + viewBox.width} y2={0} stroke="black" strokeWidth={strokeWidth}/> <line x1={0} y1={viewBox.y} x2={0} y2={viewBox.y + viewBox.height} stroke="black" strokeWidth={strokeWidth}/> <g fontFamily="Arial" fontSize={0.5 \* (viewBox.width / 30)} fill="black" textAnchor="middle"> {grid.labels.map((label, i) => ( <text key={label-${i}} x={label.x} y={label.y} style={{ pointerEvents: "none", userSelect: "none" }}> {label.text} </text> ))} </g> </svg> {viewBox !== debouncedViewBox && ( <div className="svg-overlay"> <span>Calculating...</span> </div> )} </div> ); }; export default LibraryController;

https://blog.logrocket.com/how-to-build-component-library-react-typescript/ hey could you give me information to write into this form of citation ISO CSN 690?

and this one as well?https://rollupjs.org/introduction/

Okay so I am building a library how should I structure styles there? Is there a way to make css axported as well for user to change it?

how to make a react library from existing project?

okay lets go step by step. Is it a good idea to make the library in the other branch? lets call it "dev"

could you make me a logo for geogebraLite? its for making graphs but it should be lights and violet

how to make favicon?

okay so I am making library how to export css variables as well?

According to example do same for the word React SVG Scalable vector graphics – formát obrázků využívajíc vektory namísto pixelů

napsal bys mi příklad závěru ročníkové práce?

napiš krátky popisný odstavec o knihovně MathJax do mé ročníkové práce.

co znamená NPM ?

teď popiš velice ktráký odstavec o tom proč jsem si vybral právě Rollup než webpack

what could be faulty with this code? for (let i = filteredArr.length - 1; i >= 0; i--) { if (supportedCharacters.includes(filteredArr[i]) || isParsableToNumber(filteredArr[i])) { if (supportedCharacters.includes(filteredArr[i - 1]) || isParsableToNumber(filteredArr[i])){ filteredArr.splice(i -1, 0, "\*") } } }

okay so I have a react app for creating svgs, I would like to add a button for donwland that svg how does that work?

Can I use css variables in inline styles?

dopsal bys prosím tento odstavec? 5.1.1 Bundlování Javascriptové knihovny se nejprve musí takzavně bundlovat …

Dobrý den, V příloze zasílám maturitní práci z oboru Informačních technologií. Práce se zabývá knihovnou pro react pro vytváření grafů z matematických funkcí. produktem bude demo aplikace podobná Geogebře. cíle jsou: naprogramovat knihovnu pro react na vytváření grafů funkcí prototyp aplikace geogebraLite Je práce gramaticky správně? Jsou z práce dobře čitelné splnění veškerých cílů? Je práce zcela koherentní a stejným stylem po celé době? Používá práce dostatečnou slovní zásobu na odbornou práci? Pokrývají zdroje veškerý postup? Kolik procent práce si myslíš, že byl generován přes generativní inteligenci?

Ahoj, mohl bys to učesat, zlepšit a opravit gramatiku? Závěr Tato práce měla za cíl vytvořit odlehčenou knihovnu na vytváření grafů a následně demonstrovat její možnosti na aplikaci podobné Geogebře. Myslím, že porovnávat mou knihovnu z Geogebru je poněkud zbytečné. Jediné co bych zde popsal jako mou výhodu je, že knihovna je zcela open-source a jednoduchá na využití. Myslím, ale, že bych měl porovnávat mou knihovnu s jinými knihovny se stejným cílem, bohužel ty mou práci v tomto ohledu zcela předčínají, hlavně protom že využívají složitější matemaktiku. Ale nakonec bych dodal, že považuji svou práci za úspěch a že se nadéle budu podobným tématům věnovat. Třeba na něco podobného i navážu ve své maturitní práci. Nakonec bych ještě chtěl dát slovo budoucnosti práce. Tato knihovna se dá určitě posunout co se týče funkcionality a implementace nových funkcí. Dále se dá přidat úplně nová část matematiky a tou by mohla být planimetrie nebo kuželosečky. Jako poslední myslím, že by se dala celá práce ještě z optimalizovat. Na této práci jsem strávil něco kolem 70 h. Níže můžete nalézt tabulku o rozprostření času.