**---------------------------**

**- I. Truss Data Structure -**

**---------------------------**

**TrussInfo object**

**----------------**

\* var output = {'containsTruss': false, 'trussStrokeIds': [] };

\* {boolean} containsTruss - true if there is a truss, false otherwise

\* {string[]} trussStrokeIds - contains an array of stroke IDs that belong to the truss in the target sketch, or none if there is no truss in the sketch

**-------------------------------**

**- II. Standard Data Structure -**

**-------------------------------**

**Standard Sketch Object**

**----------------------**

\* \_id

\* time

\* id

\* domain

\* shapes

\* strokes

\* substrokes

\* points

**Standard Stroke Object**

**----------------------**

\* {string} time - the time in (number-parsable) string format

\* {string[]} points - the array of point IDs in UUIN format

- "points" should actually be called "pointIds" since they are just the point's IDs (no point data)

\* {string} id - the stroke's ID in UUIN

\* Example:

var stroke = {time: "1327816353582", points: Array(151), id: "00000000-3859-4941-8000-00726df3f369"}

**Standard Stroke.Points Object**

**-----------------------------**

\* {string} <point\_id> - there are X number of properties that are named as their point ID, and their value is the corresponding point data

\* Example:

var point = stroke.points["00000000-3859-4941-8000-00726df3f369"];

console.log(point); // {x: 374, y: 110, time: "1327816344243"}

**------------------------------**

**- III. Compact Sketch Object -**

**------------------------------**

**Compact Sketch Object Conversion**

**--------------------------------**

To convert a standard sketch object to a compact sketch object, follow the steps below:

1. Include the sketchio.js file.

2. Add the following line between <head> and </head> in your HTML file: <script src="sketchio.js"></script>

3. Inside the checkSketchForTruss(sketch) function, include the following line near the beginning: sketch = SketchIo.simplify(sketch);

Your sketch object should now be a compact sketch object. The simplify(...) method in the sketchio.js file's SketchIo namespace both cleans and compacts the original standard sketch object format. Note the following:

\* compact sketch object is null ==> assume that there are no trusses, since cleaned sketch is empty

\* compact sketch object is non-null ==> code as usual

**Compact Sketch Object**

**---------------------**

\* {string} id

\* {number} time

\* {Stroke[]} strokes

**Compact Stroke Object**

**---------------------**

\* {string} id

\* {number} time

\* {Point[]} points

**Compact Point Object**

**--------------------**

\* {number} x

\* {number} y

\* {number} time

\* {string} id

**--------------------------**

**- IV. SketchIo Namespace -**

**--------------------------**

**compact(...) function**

**---------------------**

Input: raw standard sketch object

Output: clean compact sketch object

**Error-Handling Checks**

**---------------------**

Check #0: check if input exists

\* input is null ==> return null

\* input is undefined ==> return null

Check #1: check if input has shapes property

\* input.shapes is null ==> return null

\* input.shapes is undefined ==> return null

\* input.shapes is empty ==> return null

Check #2: check if input.shapes[0] has interpretation property

\* input.shapes[0].interpretation does not exist ==> return null

Check #3: check if input has strokes property

\* input.strokes does not exist ==> return null

Check #4: check if input.strokes exist

\* input.strokes is null ==> return null

\* input.strokes is undefined ==> return null

\* input.strokes is empty ==> return null

Check #5: check if input has points property

\* input.points does not exist ==> return null

Note: stroke = strokes[i], for 0 <= i < strokes.length

Check #6: check if stroke exists

\* stroke is null ==> skip stroke

\* stroke is undefined ==> skip stroke

Check #7: check if stroke has points property

\* stroke.points does not exist ==> skip stroke

Check #8: check if stroke.points exist

\* stroke.points is empty ==> skip stroke

Check #9: check if each point IDs in input.strokes maps to an actual point in input.points

\* point ID in input.strokes[i].points does not map to a point in input.points ==> skip point

Check #10: check if point has x property

\* point does not have x property ==> skip point

Check #11: check if point has numerical x-value

\* point does not have x time ==> skip point

Check #12: check if point has y property

\* point does not have y property ==> skip point

Check #13: check if point has numerical y-value

\* point does not have y time ==> skip point

Check #14: check if point has time property

\* point does not have time property ==> skip point

Check #15: check if point has numerical time-value

\* point does not have numerical time ==> skip point

Check #16: check if there are any points after cleaning

\* stroke's points are empty after cleaning ==> skip stroke

Check #17: check if there are any strokes after cleaning

\* strokes is empty after cleaning ==> return null

Check #18: check if sketch has any strokes after compacting

\* sketch is empty after compacting ==> return null