# LINKED BUILDING DATA MEETS IFC.JS

## **ABOUT US**

Antonio González Viegas



Mads Holten Rasmussen



## IFC.JS

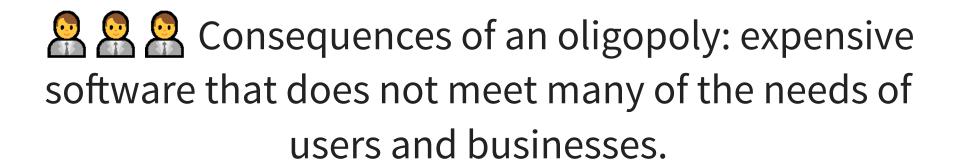
An open source library for IFC on the web

## WHY?

BIM is hard, thus only the big players can compete in equal terms.

## WHY?

## WHY?



### CONCEPT

Anyone can create a videogame for free in some hours.



>> What if we could make a BIM tool in an afternoon and completely free?

```
import { IfcViewerAPI } from 'web-ifc-viewer';
4 const container = document.getElementById('viewer-container
  const viewer = new IfcViewerAPI({ container });
8 viewer.addAxes();
9 viewer.addGrid();
10
   const input = document.getElementById("file-input");
14 input.addEventListener("change",
```

```
import { IfcViewerAPI } from 'web-ifc-viewer';
8 viewer.addAxes();
```

```
const container = document.getElementById('viewer-container
  const viewer = new IfcViewerAPI({ container });
8 viewer.addAxes();
```

```
8 viewer.addAxes();
9 viewer.addGrid();
```

```
viewer.addAxes();
   const input = document.getElementById("file-input");
13
   input.addEventListener("change",
15
     async (changed) => {
     const file = changed.target.files[0];
16
     const ifcURL = URL.createObjectURL(file);
18
     viewer.IFC.loadIfcUrl(ifcURL);
19
     },
20
     false
21 );
```

#### Example 1: How it works

#### Example 2: Making a BIM tool inside Twitter

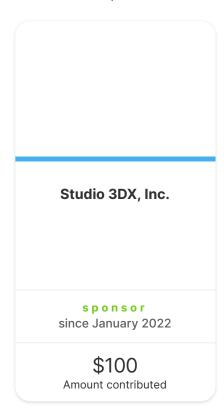


## Example 3: BIM + GIS in 48 hours

#### **EXAMPLE 4: DESKTOP APPLICATION**

#### **SUSTAINABILITY**

#### 1 organization is supporting IFC.js Contribute on Open Collective



17 individuals are supporting IFC.js Contribute on Open Collective

## IFC-LBD

An open source converter built in IFC.js

- Includes a CLI tool
- Works with web-ifc-three

```
const rdf = await lbdParser.parseBOTTriples(ifcApi, mod
```

- Currently supports BOT, Products and FSO
- Modular and easy to extend
- Written in TypeScript

## **USE AS A CLI TOOL**

npm install -g ifc-lbd

Recorded with asciinema

Recorded with asciinema

```
2 import * as WebIFC from "web-ifc/web-ifc-api.js";
   import { LBDParser } from "ifc-lbd";
 5 import * as path from 'path';
  import { readFile } from "fs";
   import * as util from "util";
  const readFileAsync = util.promisify(readFile);
   async function main(){
const modelPath = path.join( dirname, './Duplex.ifc');
    const modelData = readFileAsync(modelPath);
14
```

```
async function main(){
    // Read file
     const modelPath = path.join( dirname, './Duplex.ifc');
13
14
     const modelData = readFileAsync(modelPath);
```

```
16
     // Init API and load model
     const ifcApi = new WebIFC.IfcAPI();
     await ifcApi.Init();
18
     const modelID = ifcApi.OpenModel(ifcModelData);
```

```
21
22
     const lbdParser = new LBDParser();
23
     const botTriples = await lbdParser.parseBOTTriples(ifcApi
24
     console.log(botTriples);
```

## DEEPER LOOK INTO THE CODE

#### PATH SEARCH HELPERS

- buildRelOneToOne()
- buildRelOneToMany()

# **BUILDRELONETOONE()**

For example: IFCRELSPACEBOUNDARY

```
1 function buildRelOneToOne(
2    ifcAPI: WebIFC.IfcAPI, modelID: number = 0,
3    relationshipType: number, subjectRef: string,
4    targetRef: string, rdfRelationship: string,
5    includeInterface: boolean = false,
6    biderectional: boolean = false)
```

# **BUILDRELONETOONE()**

#### SPACE/ELEMENT ADJACENCY

```
private async buildSpaceAdjacentElementRelationships() {
     const subjectRef = "RelatingSpace";
     const targetRef = "RelatedBuildingElement";
     const rdfRelationship = "bot:adjacentElement";
     return await buildRelOneToOne(
       this.ifcAPI, // API
       this.modelID, // modelID
       IFCRELSPACEBOUNDARY, // relationship type
       subjectRef, // what rel is used to find subject of trip
       targetRef, // what rel is used to find object of triple
       rdfRelationship, // what's the predicate of the new tri
       true // Add triple in both directions?
13
     );
```

# **BUILDRELONETOMANY()**

For example: IFCRELCONTAINEDINSPATIALSTRUCTURE

```
1 function buildRelOneToMany(
2    ifcAPI: WebIFC.IfcAPI, modelID: number = 0,
3    relationshipType: number, subjectRef: string,
4    targetRef: string, rdfRelationship: string,
5    subjectClassConstraint?: number,
6    targetClassConstraint?: number)
```

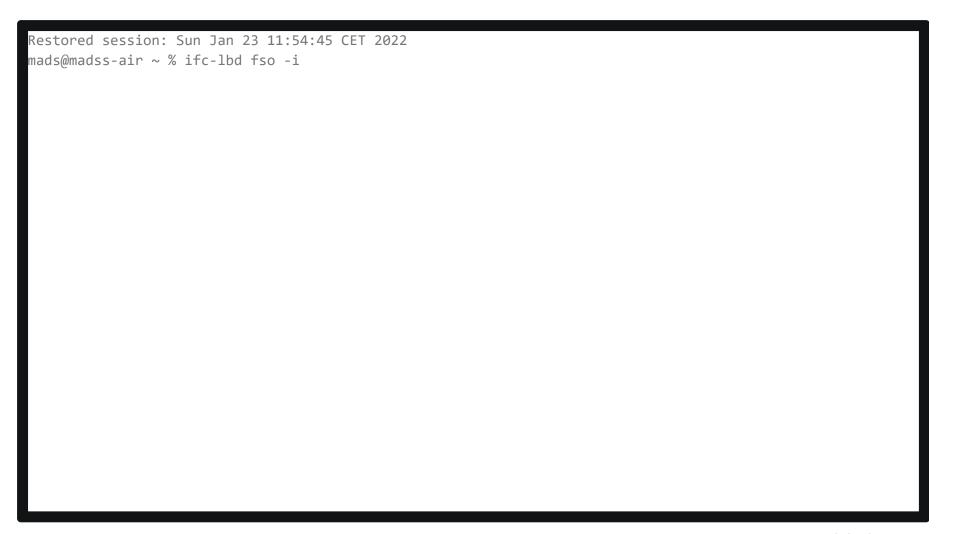
# **BUILDRELONETOONE()**

#### SPACE/ELEMENT ADJACENCY

```
private async buildSpaceContainedElementRelationships() {
  const subjectRef = "RelatingStructure";
  const targetRef = "RelatedElements";
  const rdfRelationship = "bot:containsElement";
  const subjectClass = IFCSPACE;
  return await buildRelOneToMany(
    this.ifcAPI, // API
    this.modelID, // modelID
    IFCRELCONTAINEDINSPATIALSTRUCTURE, // relationship type
    subjectRef, // what rel is used to find subject of trip
    targetRef, // what rel is used to find objects of tripl
    rdfRelationship, // what's the predicate of the new tri
    subjectClass // subjects must be of this type
  );
```

# POST-PROCESSING WITH N3 + COMUNICA

FSO uses some pattern matches that require postprocessing with SPARQL update queries



- 1. Load data into N3 store
- 2. Use Comunica to query the store
- 3. Serialize store content to file with N3 stream writer

```
async function componentConections(): Promise<void>{
     const query =
       PREFIX fso: <https://w3id.org/fso#>
       INSERT {
 5
         ?el fso:connectedWith ?e2.
         ?e2 fso:connectedWith ?e1.
         ?e1 fso:feedsFluidTo ?e2 .
         ?e2 fso:hasFluidFedBy ?e1
       WHERE {
11
         ?el fso:connectedPort ?pl .
12
         ?p1 fso:connectedPort ?p2 .
13
         ?p2 fso:connectedComponent ?e2 .
         ?p1 a fso:OutPort .
```

```
<https://w3id.org/fso#>
       PREFIX fso:
       INSERT {
         ?el fso:connectedWith ?e2 .
         ?e2 fso:connectedWith ?e1.
         ?e1 fso:feedsFluidTo ?e2 .
         ?e2 fso:hasFluidFedBy ?e1
10
       WHERE {
11
         ?el fso:connectedPort ?pl .
         ?p1 fso:connectedPort ?p2 .
13
         ?p2 fso:connectedComponent ?e2 .
14
         ?p1 a fso:OutPort .
15
         ?p2 a fso:InPort .
16
         t this execute[IndateOuery(query):
```

```
await this.executeUpdateQuery(query);
```

```
async function segmentLengths(): Promise<void>{
     const query = `INSERT{
       ?seg ex:length ?d
 5
     WHERE {
       ?seq a fso:Segment ;
         fso:connectedPort ?port1 , ?port2 .
       FILTER(?port1 != ?port2)
       ?port1 omg:hasGeometry/fog::asSfa v2-wkt ?p1 .
       ?port2 omg:hasGeometry/fog::asSfa v2-wkt ?p2 .
11
       BIND(geosf:distance(?p1, ?p2, 3) AS ?d)
13
     await this.executeUpdateQuery(query);
14 }
```

```
async function segmentLengths(): Promise<void>{
       ?seg ex:length ?d
11
       BIND(geosf:distance(?p1, ?p2, 3) AS ?d)
```

```
'http://www.opengis.net/def/function/geospargl/distance'(ar
     // Number of decimals? (default = 8)
     const decimals = args[2] != undefined ? parseFloat(args[2])
     if (args[0].termType === 'Literal' && args[1].termType ==
10
       const p1 = parseWKT(args[0].value);
12
       const p2 = parseWKT(args[1].value);
14
       const a = p1[0] - p2[0];
```

```
// Number of decimals? (default = 8)
const decimals = args[2] != undefined ? parseFloat(args[2])
```

```
11
       const p1 = parseWKT(args[0].value);
12
       const p2 = parseWKT(args[1].value);
14
       const a = p1[0] - p2[0];
15
       const b = p1[1] - p2[1];
16
       const c = p1[2] - p2[2];
18
       const d: number = round(Math.sqrt(a * a + b * b + c * c
```

```
20
       return DF.literal(d.toString(), DF.namedNode('http://ww
```

```
24
25
     return DF.literal("ERROR");
```

# PLEASE JOIN THE EFFORT IN FURTHER DEVELOPING THESE OPEN SOURCE PROJECTS!

- IFC-LBD
- Comunica geoSPARQL