b3dm 头部

magic: char[4] 4bytes

version: uint32 4bytes

byteLength: uint32 4bytes

featureTableJSONByteLength: uint32 4bytes

featureTableBinaryByteLength: uint32 4bytes

batchTableJSONByteLength: uint32 4bytes

batchTableBinaryByteLength: uint32 4bytes

28 Bytes

b3dm 数据体 size=(b3dm头部.byteLength – 28)bytes

featureTable

JSON头部：ftJSON
大小：b3dmHeader.featureTableJSONByteLength]

二进制数据体：ftBinary
大小：
b3dmHeader.featureTableBinaryByteLength

batchTable[可以没有]

JSON头部：btJSON
大小：b3dmHeader.batchTableJSONByteLength]

二进制数据体：btBinary
大小：
b3dmHeader.batchTableBinaryByteLength

glb

b3dm.body.batchTable

头部：batchTableJSON

大小：b3dm.header.featureTableJSONByteLength

格式之一
batchid { byteOffset + componentType + type }
byteOffset：此id对于btBinary的0字节偏移量
componentType：每个batchid中的值的类型，例如FLOAT占4byte
type: 每个batchid的组合类型，例如VEC3有3个值/组

必须是8byte的倍数(64bit)，如果不够，用ASCII的空格字符在末尾补齐（0x20, 1byte）

二进制数据体batchTableBinary

大小：b3dm.header.featureTableBinaryByteLength

布局为每个batchid的连续排列；
大小也等于每个batchid的大小之和，见右边例子。

必须是8byte倍数，暂时不确定补齐和偏移量

```
{
    "height" : {
        "byteOffset" : 0,
        "componentType" : "FLOAT",
        "type" : "SCALAR"
    },
    "geographic" : {
        "byteOffset" : 40,
        "componentType" : "DOUBLE",
        "type" : "VEC3"
    },
}
```

batchTableJSON

两个batchid：height、geographic

batchTableBinary

Height 1 value	Height 2 value	...	Height 10 value
FLOAT 4bytes	FLOAT 4bytes		FLOAT 4bytes
SCALAR 1个值/组	SCALAR 1个值/组		SCALAR 1个值/组

batchLength = 10个，总计4bytes*1个值/组*10个=40bytes

geographic 1 [value1, value2, value3]	...	geographic 10 [value1, value2, value3]
DOUBLE 8bytes		DOUBLE 8bytes
VEC3 3个		VEC3 3个

batchLength = 10个，总计8bytes*3个值/组*10个=240bytes

总计 280bytes=40+240=b3dm.header.batchTableBinaryByteLength