Differences between DOM and SAX. When to use what?

Before going through the differences, if you need a refresh of what SAX and DOM are, please refer to this article - SAX, DOM, JAXP, & JDOM >>.

While comparing two entities, we tend to see both of them as competitors and consequently comparing them to find a winner. This of course is not applicable in every case - not at least in the case of SAX and DOM. Both have their own pros and cons and they are certainly not in direct competition with each other.

SAX v/s DOM

Main differences between SAX and DOM, which are the two most popular APIs for processing XML documents in Java, are:-

- **Read v/s Read/Write:** SAX can be used only for reading XML documents and not for the manipulation of the underlying XML data whereas DOM can be used for both read and write of the data in an XML document.
- **Sequential Access v/s Random Access:** SAX can be used only for a sequential processing of an XML document whereas DOM can be used for a random processing of XML docs. So what to do if you want a random access to the underlying XML data while using SAX? You got to store and manage that information so that you can retrieve it when you need.
- **Call back v/s Tree:** SAX uses call back mechanism and uses event-streams to read chunks of XML data into the memory in a sequential manner whereas DOM uses a tree representation of the underlying XML document and facilitates random access/manipulation of the underlying XML data.
- **XML-Dev mailing list v/s W3C:** SAX was developed by the XML-Dev mailing list whereas DOM was developed by W3C (World Wide Web Consortium).
- **Information Set:** SAX doesn't retain all the info of the underlying XML document such as comments whereas DOM retains almost all the info. New versions of SAX are trying to extend their coverage of information.

Usual Misconceptions

- **SAX is always faster:** this is a very common misunderstanding and one should be aware that SAX may not always be faster because it might not enjoy the storage-size advantage in every case due to the cost of call backs depending upon the particular situation, SAX is being used in.
- **DOM always keeps the whole XML doc in memory:** it's not always true. DOM implementations not only vary in their code size and performance, but also in their memory requirements and few of them don't keep the entire XML doc in memory all the time. Otherwise, processing/manipulation of very large XML docs may virtually become impossible using DOM, which is of course not the case.

How to choose one between the two?

It primarily depends upon the requirement. If the underlying XML data requires manipulation then almost always DOM will be used as SAX doesn't allow that. Similarly if the nature of access is random (for example, if you need contextual info at every stage) then DOM will be the way to go in most of the cases. But, if the XML document is only required to be read and that too sequentially, then SAX will probably be a better alternative in most of the cases. SAX was developed mainly for pasring XML documents and it's certainly good at it. SO, if you need to process an XML document maybe to update a datasource, SAX will probably make a alternative.

Requirements may certainly fall between the two extremes discussed above and for any such situation you should weight both the alternatives before picking any of the two. There are applications where a combination of both SAX and DOM are used for XML processing so that might also be an alternative in your case. But, basically it would be a design decision and evidently it would require a thorough analysis of the pros and cons of all possible approaches in that situation.