1. In the demo, for the encoded byte stream, we were printing the corresponding decimal numbers rather than hex codes. In case, you want to see the hex codes, you can simply pass the byte array to the following method, which returns a String. The class is from the package javax.xml.bind

DatatypeConverter.printHexBinary(byte[])

Resource: http://stackoverflow.com/questions/9655181/how-to-convert-a-byte-array-to-a-hex-string-in-java

2. Below is an article by Joel Spolsky, who is one of the creators of Stackoverflow. The article is titled as "The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!)". It is a very popular article and is a must read. We must have covered lot of its details, but you can read it when you get a chance.

http://www.joelonsoftware.com/articles/Unicode.html

3. Below is another great resource on Unicode. Do check it out when you get a chance.

http://unicode.org/faq/utf_bom.html

4. Below is a great answer on Unicode & UTF. Another must read when you get a chance. We must have covered most of the stuff, but it is an enjoyable read!!

http://stackoverflow.com/questions/2241348/what-is-unicode-utf-8-utf-16?rq=1

5. Below is also a nice answer by Jon Skeet on stackoverflow site. Here is what he says about UTF16 benefit:

"*If you're happy ignoring surrogate pairs (or equivalently, the possibility of your app needing characters outside the basic multilingual plane), UTF-16 has some nice properties, basically due to the size per code unit being constant. You know how much space to allocate for a given number of code units, and you can index directly into that space to access the nth code unit. Those aren't usually important aspects for a text file - although they certainly are if you want to use random access - but size generally is important for text files*."

What he says is because size per code unit is constant,

- You can quickly know much space is need to allocate for a given string
- Next, he says it would be very fast to use a method like String.charAt(index)

http://stackoverflow.com/questions/14942092/why-does-net-uses-the-utf16-encoding-for-string-but-uses-utf8-as-default-for

6. You can access www.unicode-table.com to search for characters and their corresponding code points and UTF representations. You can also search by code points, character names, etc. For example below is the url for tears of joy Emoji. Great resource.

http://unicode-table.com/en/1F602/

7. Some additional notes on ASCII.

- American Standard Code for Information Interchange
- Uses 7-bit code to represent each character
- Codes 41H – 5AH (H means hexadecimal) represents 'A' to 'Z'
- Codes 61H – 7AH represents 'a' to 'z'
- Codes 30H – 39H represents '0' to '9'
- Codes 20H – 7EH are printable (displayable) characters
- Codes 00H to 7FH are special control characters, which are non-printable (non-displayable), e.g., 07H for beep
- So, only first 128 codes were used by ASCII as it was based on 7 bits
- Soon, different groups realized that they have room for up to eights bits as ASCII is only 7 bits
- So, they used codes from 128 – 255 in different ways independently

Other Interesting Resources:

- Unicode surrogate programming with the Java language
  http://www.ibm.com/developerworks/library/j-unicode/