Ten Things About Immutable Collections

(In progress. rough outline.)

1. Memory efficiency. Without mutability, there's no need to leave space for elements that might be added later. Immutable collections are always more compact than mutable equivalents.
2. Order is preserved. Every immutable collection (except the sorted ones, of course) iterates over elements in the order they were added to the builder.
3. copyOf() short-circuiting. You can always call ImmutableXXX.copyOf defensively on collections passed to your methods, and it's smart enough that if the collection is already immutable, and the collection isn't a partial view of a larger collection, it won't actually do the copying.
4. They're types, not implementations – think of them like interfaces.
5. ImmutableCollections have an asList() view
6. They don't like null.
7. How they're better than unmodifiable()
8. ImmutableList has a reverse() view
9. They have builders
10. They don't try to protect you from your equals() or hashCode() method being slow.
11. As with all immutable objects, no thread-safety concerns (as long as the contents are thread safe)