* Learn [to think functionally](http://www.yuiblog.com/blog/2010/02/24/video-crockonjs-3/) ( *because that's where the power is* )
* Use the [The Good Parts](http://rads.stackoverflow.com/amzn/click/0596517742) ( *to sleep well* )
* Use [Jslint](http://jslint.com/) ( *to validate* )
* Avoid browser sniffing, and use [feature detection](http://peter.michaux.ca/articles/feature-detection-state-of-the-art-browser-scripting) instead ( *because browsers lie* )
* Keep the [global scope clear](http://yuiblog.com/blog/2006/06/01/global-domination/) ( *to avoid collisions* )
* [Namespace](http://peter.michaux.ca/articles/javascript-namespacing) your code ( *to avoid collisions* )
* [Don't mess with host objects](http://perfectionkills.com/whats-wrong-with-extending-the-dom/) ( *because they can't be trusted* )
* Consider [ES5 Strict](http://ejohn.org/blog/ecmascript-5-strict-mode-json-and-more/) mode ( *because it is the future* )
* Follow the [Top-Rated bloggers](http://stackoverflow.com/questions/409056/top-rated-javascript-blogs) ( *because they know "stuff"* )

Bonus: avoid gotchas

**Gotchas**

* using for.. in for arrays (use simple loop, because Array properties can interfere)
* relying on global undefined, which can be modified (use typeof instead or define your own)
* not using *var* for local variables
* using more than one *var* for the same variable (can be confusing, error prone)
* checking "if (somevar)" to see if it's defined, when *somevar* can be 0, false, or "" (empty string)
* polluting global scope (possible collisions)
* misusing closures and causing memory leaks (release event listeners!)
* using browser sniffing (use feature detection instead)
* augmenting host objects (don't do that, they cannot be trusted)
* augmenting native objects too much (anyone can overwrite your functions)
* not using built-in functions (learn the language!)
* forgetting to use *new* for constuctor functions (convention: capitalize first letter)
* typing/see *new*, and think: I know this stuff from somewhere (it's not Java)
* unconsciously initializing variables to null (it's not C/C++ either)
* not realizing that functions are objects (yes they can have properties)
* relying too much on floating point arithmetic (check: 0.1 + 0.2 == 0.3)
* not realizing the difference between == v. === ( the last one is type safe, use == if you know it well)
* not realizing that arguments are passed by reference (except primitive types)
* changing the arguments array
* relying on semicolon insertion

<http://javapapers.com/core-java/java-features-and-history/>

<http://javapapers.com/core-java/externalizable-vs-serializable/>