From : colorForth Home Page https://www.inventio.co.uk/colorForth_archive/Chuck_Moore_Website/binding.html

Early Binding

The interactive nature of Forth leads to recognition of 3 'times':

- Edit time. The programmer is editing source code.
- Compile time. Source code is being interpreted and compiled.
- Run time. Compiled code is being executed.

The earlier in this Edit/Compile/Run sequence a concept can be resolved, the more efficient the code. It is better to do arithmetic at Compile time, than repeat the calculation every Run time. It is better to factor code at Edit time than to expect a smart compiler to do so every Compile time.

Floating-point

Floating-point arithmetic is a classic example of this. It is easy to implement floating-point in Forth, either hardware or software. I, for one, rarely do so.

Fixed-point add is blindingly fast, with no shifting overhead. Fixed-point multiply avoids normalize and rounding cost. Multiplying by a ratio (*/ operator) accurately rescales integers. None of these needs FIX or FLOAT operators, or has to move data to a different stack. The only drawback is the need to choose units and use them consistently. For example, my VLSI simulator uses mV, uA, fC and uK. Multiplying and dividing these takes a little thought as to scale factors. This thought is done at Edit time!

Floating-point would probably use V, A, C and K. These can be combined with formulas from some book. The computer will do the scaling, but does it at Run time! A gigaflop calculation will scale the results giga times. This is convenient and reasonable for prototyping. But insanity for production code.

To repeat: resolve units early in the Edit/Compile/Run sequence.

Conditionals

Another example concerns conditional statements - IF . . . THEN. Minimize them! A conditional is obviously executed at Run time. Let the programmer make the decision at Edit time. This might reduce the range or versatility of the code, but it also reduces testing and improves speed and reliability.

An alternative is to use words like MAX, MIN, ABS that have conditionals built in. They're simple and have been thoroughly tested.

Avoid like the plague, conditional compilation - conditionals at Compile time. These lead to code extremely difficult to read. And very easy to compile incorrectly. For different situations, simply have different versions of the program. In Forth, this usually means blocks with different LOAD sequences.

Portability

Don't try for platform portability. Most platform differences concern hardware interfaces. These are intrinsically different. Any attempt to make them appear the same achieves the lowest common denominator. That is, ignores the features that made the hardware attractive in the first place.

Achieve portability by factoring out code that is identical. Accept that different systems will be different.