Regular expressions are widely used, but they are inherently hard to understand and (re)use, which is primarily due to the lack of abstraction mechanisms that causes regular expressions to grow large very quickly. The problems with understandability and usability are further compounded by the viscosity, redundancy, and terseness of the notation. As a consequence, many different regular expressions for the same problem are floating around, many of them erroneous, making it quite difficult to find and use the right regular expression for a particular problem. Due to the ubiquitous use of regular expressions, the lack of understandability and usability becomes a serious software engineering problem. In this paper we present a range of independent, complementary representations that can serve as explanations of regular expressions. We provide methods to compute those representations, and we describe how these methods and the constructed explanations can be employed in a variety of usage scenarios. In addition to aiding understanding, some of the representations can also help identify faults in regular expressions. Our evaluation shows that our methods are widely applicable and can thus have a significant impact in improving the practice of software engineering.