Dynamic features of programming languages such as dynamic type system, dynamic method calls, dynamic code execution, and dynamic data structures provide the flexibility which can accelerate the development, but on the other hand they reduce the information that is checked at compile time and thus make programs more error-prone and less efficient. While the problem of lacking compile time checks can be partially addressed by techniques of static analysis, dynamic features pose major challenges for these techniques sacrificing their precision, soundness, and scalability. To tackle this problem, we propose a framework for static analysis that automatically resolves these features and thus allows defining sound and precise static analyses similarly as the analyzed program would not use these functions. To build the framework, we propose a novel heap analysis that models associative arrays and dynamic (prototype) objects. Next, we propose value analysis providing additional information necessary to resolve dynamic features. Finally, we propose a technique that automatically and generically combines value analysis and a heap analysis modeling associative arrays and prototype objects.