Below are few advantages of static typing:

- Static typing helps in earlier detection of programming mistakes as it allows variable types to be checked at compile time itself. With dynamic type checking, majority of type checking is performed at runtime. Due to this there are chances for programming mistakes to make their way into runtime environment. Let’s consider the following example taken from [3]. The code is written in Python, which is dynamically typed.

  ```
  def silly(a):
      if a > 0:
          print 'Hi'
      else:
          print 5 + '3'
  ```

  If the method ***silly*** is invoked with a value greater than 0, then we are good and the string ‘Hi’ gets printed. However, if 0 or a negative number is passed as input, then we get a **runtime error** as addition operation is applied on a numeric & a non-numeric value. This will not happen in Java as the statement corresponding **print 5 + ‘3’** would give a compilation error.

- Static typing is also helpful during method overloading. With method overloading, we can have multiple versions of the same method in the same class. For example, the following two methods are named as ***foo*** in the same class. This is possible because the methods accept different input data, i.e., one is accepting an integer as input while the other is accepting character data. Without static type checking, method overloading wouldn’t be possible. Method overloading will be discussed later in the chapter while discussing methods.

  ```
  void foo(int i) { … }
  void foo(char c) { … }
  ```

- Static typing also permits better developer experience in IDEs such as Eclipse. For example, once we type a class name followed by a dot operator, eclipse will automatically display a ***drop-down menu*** displaying the variables and methods that the class has. This way we can simply choose the particular variable or method without having to type-in the method or variable name. We will see this when we switch to Eclipse.
- Maintainability is another advantage. Some people believe that ***code refactoring*** is tedious in dynamically-typed languages like JavaScript. Note that code refactoring means ***restructuring*** existing code without changing its external behavior

Static vs Dynamic Typed Languages

Checkout the links in references section, which are from the Stackoverflow Website. They compare static vs dynamic typed languages and explain what they mean. Pros & cons of both types of languages are also debated.

References

[1] http://stackoverflow.com/questions/125367/dynamic-type-languages-versus-static-type-languages

[2] http://stackoverflow.com/questions/42934/what-do-people-find-so-appealing-about-dynamic-languages

[3] http://stackoverflow.com/questions/1517582/what-is-the-difference-between-statically-typed-and-dynamically-typed-languages