Class based vs. Prototype based languages

|  |  |
| --- | --- |
| Class-based (Java) | Prototype-based (JavaScript) |
| Class and instance are distinct entities. | All objects can inherit from another object. |
| Define a class with a class definition; instantiate a class with constructor methods. | Define and create a set of objects with constructor functions. |
| Create a single object with the new operator. | Same. |
| Construct an object hierarchy by using class definitions to define subclasses of existing classes. | Construct an object hierarchy by assigning an object as the prototype associated with a constructor function. |
| Inherit properties by following the class chain. | Inherit properties by following the prototype chain. |
| Class definition specifies *all* properties of all instances of a class. Cannot add properties dynamically at run time. | Constructor function or prototype specifies an *initial set* of properties. Can add or remove properties dynamically to individual objects or to the entire set of objects |

Notes:

**strongly typed(Java)** or **weakly typed** (**loosely typed-Javascript**). These terms do not have a precise definition, but in general, a strongly typed language is more likely to generate an error or refuse to compile if the argument passed to a function does not closely match the expected type. On the other hand, a very weakly typed language may produce unpredictable results or may perform implicit type conversion.

|  |  |
| --- | --- |
| Variable data types are not declared (**dynamic** **typing-JavaScript**). | Variable data types must be declared (**static** **typing-Java**). |

Reference: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Details_of_the_Object_Model#Class-Based_vs._Prototype-Based_Languages>