‘This’ normally refers to the object which ‘owns’ the method, but it depends on how a function is called. However “this” varies in different environments.

**Global**:

“This” in a global environment means it is defined as a global object-“global” but this is true only in node.

**Function**:

The value of “this” inside a function is usually defined by the function’s call. So, “this” can have different values inside it for each execution of the function.

**Methods:**

When calling a function as a method of an object, this refers to the object, which is then known as the receiver of the function call.

**call() and apply():**

Though a function’s “this”  value is set implicitly, we can also call function with explicit “this” argument call() and apply().

**bind():**

When we pass a method as a callback to another function, there is always a risk of losing the intended receiver of the method, making the “this” argument set to the global object instead.

The bind()  method allows us to permanently tie a “this” argument to a value.   
  
**Catching “this” inside an Arrow Function:**

An arrow function uses the “this” value from its enclosing execution context, since it does have one of its own.

An arrow function permanently captures the “this” value, preventing “apply” or “call” from changing it later on.

**Classes:**

A class generally contains a “constructor”, where “this” would refer to any newly created object.

But in case of methods, “this” can also refer to any other value if the method is called as an ordinary function. And just like a method, classes can also lose track of the receiver.