JavaScript

**Class Inheritance**

**[Summary](https://javascript.info/class-inheritance" \l "summary)**

1. To extend a class: class Child extends Parent:

That means Child.prototype.\_\_proto\_\_ will be Parent.prototype, so methods are inherited.

1. When overriding a constructor:

We must call parent constructor as super() in Child constructor before using this.

1. When overriding another method:

We can use super.method() in a Child method to call Parent method.

# **Static properties and methods**

## [Summary](https://javascript.info/static-properties-methods#summary)

Static methods are used for the functionality that belongs to the class “as a whole”. It doesn’t relate to a concrete class instance.

For example, a method for comparison Article.compare(article1, article2) or a factory method Article.createTodays().

They are labeled by the word static in class declaration.

Static properties are used when we’d like to store class-level data, also not bound to an instance.

The syntax is:

# **Private and protected properties and methods**

## [Internal and external interface](https://javascript.info/private-protected-properties-methods" \l "internal-and-external-interface)

In object-oriented programming, properties and methods are split into two groups:

* Internal interface – methods and properties, accessible from other methods of the class, but not from the outside.
* External interface – methods and properties, accessible also from outside the class.

To hide an internal interface we use either protected or private properties:

* Protected fields start with \_. That’s a well-known convention, not enforced at the language level. Programmers should only access a field starting with \_ from its class and classes inheriting from it.
* Private fields start with #. JavaScript makes sure we can only access those from inside the class.

**Prototypal Inheritance** 