Diagram

Description automatically generated

* Person.prototype actually has a reference back to person with the .constructor property
* Person.prototype is the prototype of all the objects that are created with the Person constructor function
* When a new object is created with the new operator
  + **Empty object {} is created**
  + *this* keyword in constructor function call is set to the new object (inside function’s execution context) – ***this = {}***
    - that’s why the **name and birth year properties are set on the *this* keyword**, because doing so will **set the properties in the empty object**
* The new object is **linked to the constructor functions prototype property** by **adding the \_\_proto\_\_** property to the new object (this completes the prototype chain from:

object.\_\_proto\_\_ -> Person.prototype (prototype of object, prototype property of constructor function)-> Person (constructor function)

**Calling calcAge():**

* harley.calcAge() -> JS will look for the calcAge() method on the harley object, but won’t be able to find it
* So JS will then search for the method from the prototype of harley, and use the method from there
* This behaviour is **prototypal inheritance**
  + harley object inherited the calcAge method from its prototype
* 1000s of objects can be created can use prototypal inheritance to use the calcAge() method, improving code performance

**Diagram

Description automatically generatedPrototype Chain**

* All objects in JS have a prototype, so Person.prototype has a prototype itself -> **Object.prototype is the prototype of all constructor functions (object) objects in JS**
* Person.prototype has been built by the **built-in Object() constructor function**
* When an object literal is created, the object constructor function is used to create the object -> **{ }** is a shortcut for **new Object(**
* The entire series of links between the objects is call the **prototype chain** – object.prototype is at the top of the chain
* **Prototype chain** is very similar to the scope chain, where searching for properties/methods on objects moves up the scope chain if they cannot be found
  + For example, harley.hasOwnproperty(‘name’) method call is not on harley object
  + JS moves up the prototype chain to Person.prototype but still cannot find the method
  + JS moves up further to Object.prototype, where .hasOwnproperty() is a built in method to this prototype, and so it delegates the method to the harley object
  + This means all these built in methods don’t have to be defined on the created objects, improving code performance