**Integrantes: Gimenez Ollier Alejo Thomas, Cuello Guadalupe Sol**

**1. Generalización simbólica**

*(Las reglas escritas, la sintaxis y estructuras formales del lenguaje.)*

En el caso de **JavaScript orientado a objetos basado en prototipos**, las *generalizaciones simbólicas* son las **reglas y mecanismos que definen su forma de representar y manipular objetos y herencia**.

Algunos puntos clave:

* **Todo en JavaScript es un objeto o puede comportarse como tal.**  
  Los objetos se crean a partir de otros objetos (no de clases, como en Java o C++).
* **Herencia basada en prototipos:**  
  Los objetos pueden heredar propiedades directamente de otros objetos mediante la cadena de prototipos.
* **Funciones constructoras y prototype:**  
  Antes de class (ES6), la orientación a objetos se implementaba creando funciones que actuaban como "constructores".};
* El uso de this y del enlace dinámico del contexto también forma parte de las reglas del paradigma.

**2. Creencias de los profesionales**

*(Qué piensan los desarrolladores sobre el paradigma; qué se considera “mejor”.)*

Los programadores que usan JavaScript en su versión **basada en prototipos** suelen creer que:

1. **La herencia prototípica es más flexible** que la basada en clases.
   * Se pueden extender o modificar objetos dinámicamente sin necesidad de definir jerarquías rígidas.
   * Esto permite un desarrollo más ágil y adaptable, ideal para entornos web cambiantes.
2. **El lenguaje es más expresivo y dinámico.**
   * Se puede modificar el comportamiento de objetos en tiempo de ejecución.
   * Se pueden crear patrones de diseño más libres, sin las restricciones de la herencia clásica.
3. **JavaScript promueve la simplicidad conceptual.**
   * Todo es un objeto, no hay una distinción tajante entre “clase” e “instancia”.
   * Esto reduce la cantidad de abstracciones intermedias.
4. **Favorece la composición sobre la herencia.**
   * En lugar de crear largas jerarquías, los objetos se combinan (mixins, delegación) para reutilizar código de manera más flexible.
5. **La orientación a prototipos es más cercana al modelo real del lenguaje.**
   * Los desarrolladores más expertos creen que comprender los prototipos da una comprensión más profunda de cómo funciona JavaScript “por debajo” de las clases modernas (class es solo azúcar sintáctico sobre los prototipos).