**Conceptos y Paradigmas de lenguajes de Programación**

**2019**

**Práctica Nro 1**

**Historia, evolución y características de Leng. de Programación Objetivo:** ​Conocer​​la evolución de los lenguajes de programación y sus características.

**Ejercicio 1:** ​Los lenguajes de programación más representativos son:

**1951 - 1955**​: Lenguajes tipo assembly

**1956 - 1960**​: FORTRAN, ALGOL 58, ALGOL 60, LISP

**1961 - 1965**​: COBOL, ALGOL 60, SNOBOL, JOVIAL

**1966 - 1970**​: APL, FORTRAN 66, BASIC, PL/I, SIMULA 67, ALGOL-W

**1971 - 1975**​: Pascal, C, Scheme, Prolog

**1976 - 1980**​: Smalltalk, Ada, FORTRAN 77, ML

**1981 - 1985**​: Smalltalk 80, Turbo Pascal, Postscript

**1986 - 1990**​: FORTRAN 90, C++, SML

**1991 - 1995**​: TCL, PERL, HTML

**1996 - 2000**​: Java, Javascript, XML

|  |  |
| --- | --- |
| Años | Influencias y nuevas tecnologías |
| **1951 - 1955**​: | Hardware: Computadoras de tubo de vacío; Memoria de línea de retardo de mercurio.  Métodos: lenguajes tipo Assembly; conceptos fundamentales; subprogramas; estructuras de datos.  Lenguajes: Uso experimental de compiladores de expresión. |
| **1956 - 1960**​: | Hardware: Almacenamiento de cinta magnética. Memorias del núcleo. Circuitos de transistores.  Métodos: Tecnología de compilación temprana. Optimización de código. Intérpretes. Métodos de almacenamiento dinámico y procesamiento de listas. Gramática BNF.  Lenguajes: FORTRAN, ALGOL 58, ALGOL 60, LISP. |
| **1961 - 1965**​: | Hardware: Familias de arquitecturas compatibles; almacenamiento en disco magnético.  Métodos: Multiprogramación de sistemas operativos; compiladores dirigidos por la sintaxis.  Lenguajes: COBOL, ALGOL 60, SNOBOL, JOVIAL. |
| **1966 - 1970**​: | Hardware: Aumentar el tamaño y la velocidad; y disminuir el costo. Circuitos integrados de microprogramación.  Métodos: Sistemas de tiempo compartidos. Optimización de compiladores. Traductor de escritura del sistema.  Lenguajes: APL, FORTRAN 66, BASIC, PL/I, SIMULA 67, ALGOL-W. |
| **1971 - 1975**​: | Hardware: Minicomputadoras. Pequeños sistemas de almacenamiento masivo. Memorias semiconductoras.  Métodos: Verificación del programa. Programación estructurada. Ingeniería de software.  Lenguajes: Pascal, C, Scheme, Prolog. |
| **1976 - 1980**​: | Hardware: Microcomputadoras. Sistemas de almacenamiento masivo. Computación distribuída.  Métodos: Abstracción de datos. semantica formal. Técnicas de programación concurrente, embebidas y en tiempo real.  Lenguajes: Smalltalk, Ada, FORTRAN 77, ML |
| **1981 - 1985**​: | Hardware: Computadoras personales. Estaciones de trabajo. Videojuegos. Redes de área local. ARPANET.  Métodos: Programación orientada a objetos. Entornos interactivos. Editores dirigidos por la sintaxis.  Lenguajes: Smalltalk 80, Turbo Pascal, Postscript |
| **1986 - 1990**​: | Hardware: Edad del microordenador. Estación de trabajo de ingeniería. Arquitecturas RISC. Internet.  Métodos: Computación cliente/servidor.  Lenguajes:FORTRAN 90, C++, SML |
| **1991 - 1995**​: | Hardware: Muy rápidos estaciones de trabajo de bajo costo y microcomputadoras. Arquitecturas masivamente paralelas. Voz, video, fax, multimedia.  Métodos: Sistemas abiertos. Marcos de entorno.  Lenguajes: TCL, PERL, HTML |
| **1996 - 2000**​: | Hardware: Las computadoras como aparatos de bajo costo. Asistentes digitales personales. Red mundial. Redes domésticas basadas en cable. Almacenamiento en disco Gigabyte.  Métodos: Comercio electrónico.  Lenguajes:Java, Javascript, XML |

Indique para cada uno de los períodos presentados cuales son las características nuevas que se incorporan y cual de ellos la incorpora.

**Ejercicio 2:** ​Escriba brevemente la historia del lenguaje de programación que eligió en la encuesta u otro de su preferencia.

**C** Es un lenguaje de programación originalmente desarrollado por Dennis Ritchie entre 1969 y 1972 en los Laboratorios Bell,2 como evolución del anterior lenguaje B, a su vez basado en BCPL. Al igual que B, es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones. Se trata de un lenguaje de tipos de datos estéticos, débilmente tipificado, de medio nivel, ya que dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos. La primera estandarización del lenguaje C fue en ANSI, con el estándar X3.159-1989. El lenguaje que define este estándar fue conocido vulgarmente como ANSI C. Posteriormente, en 1990, fue ratificado como estándar ISO (ISO/IEC 9899:1990). La adopción de este estándar es muy amplia por lo que, si los programas creados lo siguen, el código es portable entre plataformas y/o arquitecturas.

**PHP** Es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue originalmente diseñado en Perl, con base en la escritura de un grupo de CGI binarios escritos en el lenguaje C por el programador danés-canadiense Rasmus Lerdorf en el año 1994 para mostrar su currículum vítae y guardar ciertos datos, como la cantidad de tráfico que su página web recib´ıa. El 8 de junio de 1995 fue publicado ”Personal Home Page Tools” después de que Lerdorf lo combinara con su propio Form Interpreter para crear PHP/FI. Dos programadores israelíes del Technion, Zeev Suraski y Andi Gutmans, reescribieron el analizador sintáctico (parser, en inglés) en 1997 y crearon la base del PHP3, y cambiaron el nombre del lenguaje por PHP: Hypertext Preprocessor. Inmediatamente comenzaron experimentaciones públicas de PHP3, y se publicó oficialmente en junio de 1998. Para 1999, Suraski y Gutmans reescribieron el código de PHP, y produjeron lo que hoy se conoce como motor Zend. También fundaron Zend Technologies en Ramat Gan, Israel. En mayo del 2000, PHP 4 se lanzó bajo el poder del motor Zend 1.0. El 13 de julio del 2007 se anunció la suspensión del soporte y desarrollo de la versión 4 de PHP,11 y, a pesar de lo anunciado, se ha liberado una nueva versión con mejoras de seguridad, la 4.4.8, publicada el 13 de enero del 2008, y posteriormente la versión 4.4.9, publicada el 7 de agosto del 2008. Según esta noticia, se le dio soporte a fallos críticos hasta el 9 de agosto del 2008. El 13 de julio del 2004, se lanzó PHP 5, utilizando el motor Zend Engine 2.0. Actualmente, se encuentra la versión 7 de PHP.

**Ejercicio 3:** ¿Qué atributos debería tener un buen lenguaje de programación? Por ejemplo, ortogonalidad, expresividad, legibilidad, simplicidad, etc. De al menos un ejemplo de un lenguaje que cumple con las características citadas.

• Claridad, sencillez y unidad (legibilidad): La sintaxis del lenguaje afecta la facilidad con la que un programa se puede escribir, por a prueba, y más tarde entender y modificar.

• Ortogonalidad: Capacidad para combinar varias características de un lenguaje en todas las combinaciones posibles, de manera que todas ellas tengan significado.

• Naturalidad para la aplicación: La sintaxis del programa debe permitir que la estructura del programa refleje la estructura lógica subyacente.

• Apoyo para la abstracción: Una parte importante de la tarea del programador es proyectar las abstracciones adecuadas para la solución del problema y luego implementar esas abstracciones empleando las capacidades más primitivas que provee el lenguaje de programación mismo. • Facilidad para verificar programas: La sencillez de la estructura semántica y sintáctica ayuda a simplificar la verificación de programas.

• Entorno de programación: Facilita el trabajo con un lenguaje técnicamente débil en comparación con un lenguaje más fuerte con poco apoyo externo.

• Portabilidad de programas.

• Costo de uso:

– Costo de ejecución del programa.

– Costo de traducción de programas.

– Costo de creación, prueba y uso de programas.

– Costo de mantenimiento de los programas. ,→ Costo total del ciclo de vida.

**Ejercicio 4:** Tome uno o dos lenguajes de los que ud. Conozca y

* Describa los tipos de expresiones que se pueden escribir en el/ellos
* Describa las facilidades provistas para la organización del programa
* Indique cuáles de los atributos del ejercicio anterior posee el/los lenguaje/s elegidos y cuáles no posee, justifique en cada caso.

# Lenguajes - ​ADA

**Ejercicio 5:** ​Describa las características más relevantes de Ada, referida a:

* Tipos de datos
* Tipos abstractos de datos – paquetes
* Estructuras de datos
* Manejo de excepciones
* Manejo de concurrencia

# Lenguajes - JAVA

**Ejercicio 6:** ​Diga para qué fue, básicamente, creado Java.¿Qué cambios le introdujo a la Web? ¿Java es un lenguaje dependiente de la plataforma en dónde se ejecuta? ¿Porqué?

**Ejercicio 7:** ​¿Sobre cuales lenguaje está basado?

**Ejercicio 8:** ​¿Qué son los applets? ¿Qué son los servlets?

Es un lenguaje de programaci´on de prop´osito general, concurrente, orientado a objetos que fue dise˜nado espec´ıficamente para tener tan pocas dependencias de implementaci´on como fuera posible. Su intenci´on es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en ingl´es como WORA, o ”write once, run anywhere”), lo que quiere decir que el c´odigo que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programaci´on m´as populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.

Influido por Pascal, C++ y Objective-C.

Definiciones: ,

→ Applets: Las applet Java son programas incrustados en otras aplicaciones, normalmente una p´agina Web que se muestra en un navegador. ,

→ Servlets: Los servlets son componentes de la parte del servidor de Java EE, encargados de generar respuestas a las peticiones recibidas de los clientes.

# Lenguajes - C

**Ejercicio 9:** ¿Cómo es la estructura de un programa escrito en C? ¿Existe anidamiento de funciones?

**Ejercicio 10:**​ Describa el manejo de expresiones que brinda el lenguaje.

Todo programa escrito en C consta de una o m´as funciones, una de las cuales se llama main. El programa siempre comenzar´a por la ejecuci´on de la funci´on main.

Cada funci´on debe contener:

→ Una cabecera de la funci´on, que consta del nombre de la funci´on, seguido de una lista opcional de argumentos encerrados con par´entesis.

→ Una lista de declaraci´on de argumentos, si se incluyen estos en la cabecera.

,→ Una sentencia compuesta, que contiene el resto de la funci´on.

No existe el anidamiento de funciones en C.

### EXPRESIONES ARITMÉTICAS

Están formadas por variables y/o constantes, y distintos operadores aritméticos e incrementales (+, -, \*, /, %, ++, --). Como se ha dicho, también se pueden emplear paréntesis de tantos niveles como se desee, y su interpretación sigue las normas aritméticas convencionales.

Por ejemplo, la solución de la ecuación de segundo grado:

http://www.juntadeandalucia.es/averroes/centros-tic/14005663/helvia/aula/archivos/repositorio/0/164/html/lenguajec/archivos_unidad4/ecuacion.jpg

se escribe, en C en la forma:

**x=(-b+sqrt((b\*b)-(4\*a\*c)))/(2\*a);**

donde, estrictamente hablando, sólo lo que está a la derecha del operador de asignación (=) es una expresión aritmética. El conjunto de la variable que está a la izquierda del signo (=), el operador de asignación, la expresión aritmética y el carácter (;) constituyen una ***sentencia***. En la expresión anterior aparece la llamada a la ***función de librería sqrt()***, que tiene como ***valor de retorno*** la raíz cuadrada de su único ***argumento***. En las expresiones se pueden introducir espacios en blanco entre operandos y operadores; por ejemplo, la expresión anterior se puede escribir también de la forma:

**x = (-b + sqrt((b \* b) - (4 \* a \* c)))/(2 \* a);**

### EXPRESIONES LÓGICAS

Los elementos con los que se forman estas expresiones son ***valores lógicos***; verdaderos (***true***, o distintos de 0) y falsos (***false***, o iguales a 0), y los ***operadores lógicos* ||, && y !**. También se pueden emplear los ***operadores relacionales*** (<, >, <=, >=, ==, !=) para producir estos valores lógicos a partir de valores numéricos. Estas expresiones equivalen siempre a un valor 1 (**true**) o a un valor 0 (***false***). Por ejemplo:

**a = ((b>c)&&(c>d))||((c==e)||(e==b));**

donde de nuevo la ***expresión lógica*** es lo que está entre el operador de asignación (=) y el (;). La variable ***a*** valdrá 1 si ***b*** es mayor que ***c*** y***c*** mayor que ***d***, ó si ***c***es igual a**e** ó e es igual a ***b***.

### EXPRESIONES GENERALES

Una de las características más importantes (y en ocasiones más difíciles de manejar) del C es su flexibilidad para combinar expresiones y operadores de distintos tipos en una expresión que se podría llamar general, aunque es una expresión absolutamente ordinaria de C.

Recuérdese que el resultado de una expresión lógica es siempre un valor numérico (un 1 ó un 0); esto permite que cualquier expresión lógica pueda aparecer como sub-expresión en una expresión aritmética. Recíprocamente, cualquier valor numérico puede ser considerado como un valor lógico: ***true*** si es distinto de 0 y ***false*** si es igual a 0. Esto permite introducir cualquier expresión aritmética como sub-expresión de una expresión lógica. Por ejemplo:

**(a - b\*2.0) && (c != d)**

A su vez, el operador de asignación (=), además de introducir un nuevo valor en la variable que figura a su izquierda, deja también este valor disponible para ser utilizado en una expresión más general. Por ejemplo, supóngase el siguiente código que inicializa a 1 las tres variables ***a***, ***b*** y ***c***:

**a = b = c = 1;**

que equivale a:

**a = (b = (c = 1));**

En realidad, lo que se ha hecho ha sido lo siguiente. En primer lugar se ha asignado un valor unidad a ***c***; el resultado de esta asignación es también un valor unidad, que está disponible para ser asignado a ***b***; a su vez el resultado de esta segunda asignación vuelve a quedar disponible y se puede asignar a la variable ***a***.

# Lenguajes - Python - RUBY - PHP

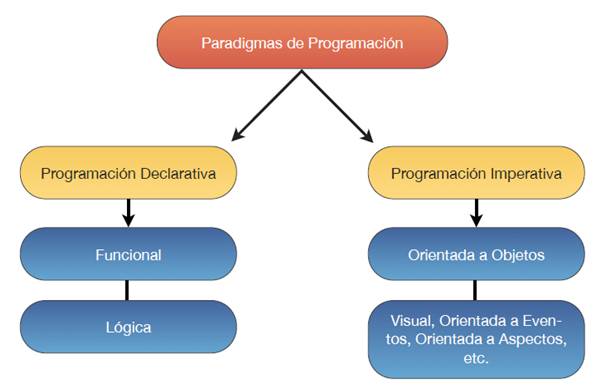
**Ejercicio 11**​: ¿Qué tipo de programas se pueden escribir con cada uno de estos lenguajes? ¿A que paradigma responde cada uno? ¿Qué características determinan la pertenencia a cada paradigma?

**Ejercicio 12:** Cite otras características importantes de Python, Ruby, PHP, Gobstone y Processing. Por ejemplo: tipado de datos, como se organizan los programas, etc.

Lenguaje Javascript

**Ejercicio 13**​: ¿A qué tipo de paradigma corresponde este lenguajes? ¿A qué tipo de Lenguaje pertenece?

**Ejercicio 14:** Cite otras características importantes de javascript. Tipado de datos, excepciones, variables, etc.



La Programación Imperativa es la más común y más conocida. Los programas imperativos son un conjunto de instrucciones que le indican a la computadora cómo realizar una tarea .Está representada por C, Basic, Fortran, etc.

La Programación Orientada a Objetos intenta simular el mundo real a través del significado de objetos que contiene características y funciones. Los lenguajes orientados a objetos se clasifican como lenguajes de quinta generación y forman parte de la programación imperativa. Se basa en la idea de un objeto, que es una combinación de variables locales y procedimientos llamados métodos que juntos conforman una entidad de programación. En la actualidad, existe una gran variedad de lenguajes de programación que soportan la orientación a objetos. Ejemplos de estos lenguajes son C++, Java, Smalltalk, Ruby on Rails, Python y JavaScript, entre muchos otros.

[Multiparadigma](https://es.wikipedia.org/w/index.php?title=Lenguaje_de_programaci%C3%B3n_multiparadigma&action=edit&redlink=1), [programación funcional](https://es.wikipedia.org/wiki/Programaci%C3%B3n_funcional),[programación basada en prototipos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_basada_en_prototipos), [imperativo](https://es.wikipedia.org/w/index.php?title=Programaci%C3%B3n_imperativa&action=edit&redlink=1), [interpretado (*scripting*)](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_interpretado).

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas Web dinámicas. Una página Web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario; a diferencia de las páginas estáticas en donde el puro lenguaje HTML sólo puede mostrar textos con estilos pero es incapaz de ofrecer interactividad.  
Técnicamente, JavaScript es un lenguaje de programación interpretado, multiplataforma, orientado a eventos o scripts , es decir, estrictamente no se considera que está orientado a objetos pero tiene la capacidad de manejarlos. Su código se incluye directamente en el mismo documento usado para el desarrollo de aplicaciones cliente-servidor en páginas HTML, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.  
A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java, pero ambas son marcas legalmente registradas de la empresa Sun Microsystems. Asimismo, es importante hacer diferencia con JScript que es la versión similar que lanzó Microsoft para competir.  
JavaScript es simple, no hace falta tener amplios conocimientos de programación. Como se mencionó, estrictamente no es un lenguaje orientado a objetos (solo maneja scripts), ya que carece de los conceptos como herencia y métodos que tienen lenguajes como C++ y Java, pero es posible definir un objeto dentro de la página Web y sobre ese objeto definir a su vez diferentes eventos que producirán la aplicación o salida deseada ofreciendo la posibilidad de crear aplicaciones “on-.line” o modificar páginas Web en tiempo real, por ejemplo, cambiar el aspecto de la página Web . Otras aplicaciones comunes son responder a eventos locales al dar clic con el mouse, realización de cálculos y validación de formularios dentro de la página, personalización, contadores de visitas página Web e inclusión de relojes y calendarios. Actualmente, todos los navegadores incluyen JavaScript y es uno de los lenguajes más populares para la Web.  
Con el surgimiento de lenguajes como PHP (Hypertext Pre-processor) del lado del servidor y JavaScript del lado del cliente, surgió AJAX (Asynchronous JavaScript And XML) que combina varias tecnologías para crear aplicaciones Web interactivas. JavaScript es considerado por muchos desarrolladores Web como la fundación para la próxima generación de aplicaciones Web dinámicas del lado del cliente.

## **Características**

Las siguientes características son comunes a todas las implementaciones que se ajustan al estándar ECMAScript, a menos que especifique explícitamente en caso contrario.

### Imperativo y estructurado

JavaScript es compatible con gran parte de la [estructura de programación](https://es.wikipedia.org/wiki/Programaci%C3%B3n_estructurada) de [C](https://es.wikipedia.org/wiki/C_(lenguaje_de_programaci%C3%B3n)) (por ejemplo, sentencias if, bucles for, sentencias switch, etc.). Con una salvedad, en parte: en C, el [ámbito](https://es.wikipedia.org/wiki/%C3%81mbito_(programaci%C3%B3n)) de las [variables](https://es.wikipedia.org/wiki/Variable_(programaci%C3%B3n)) alcanza al bloque en el cual fueron definidas; sin embargo JavaScript no es compatible con esto, puesto que el ámbito de las variables es el de la función en la cual fueron declaradas. Esto cambia con la versión de ECMAScript 2015, ya que añade compatibilidad con block scoping por medio de la palabra clave let. Como en C, JavaScript hace distinción entre [expresiones](https://es.wikipedia.org/wiki/Expresi%C3%B3n_(inform%C3%A1tica)) y sentencias. Una diferencia sintáctica con respecto a C es la inserción automática de punto y coma, es decir, en JavaScript los puntos y coma que finalizan una sentencia pueden ser omitidos.[25](https://es.wikipedia.org/wiki/JavaScript#cite_note-FOOTNOTEFlanagan200616-25)​

### Dinámico

**Tipado dinámico**

Como en la mayoría de [lenguajes de scripting](https://es.wikipedia.org/wiki/Script), el [tipo](https://es.wikipedia.org/wiki/Sistema_de_tipos) está asociado al valor, no a la variable. Por ejemplo, una variable x en un momento dado puede estar ligada a un número y más adelante, religada a una [cadena](https://es.wikipedia.org/wiki/String). JavaScript es compatible con varias formas de comprobar el tipo de un objeto, incluyendo [duck typing](https://es.wikipedia.org/wiki/Duck_typing" \o "Duck typing).[26](https://es.wikipedia.org/wiki/JavaScript#cite_note-FOOTNOTEFlanagan2006176%E2%80%93178-26)​ Una forma de saberlo es por medio de la palabra clave typeof.

**Objetual**

JavaScript está formado casi en su totalidad por [objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos). Los objetos en JavaScript son [arrays asociativos](https://es.wikipedia.org/wiki/Vector_asociativo" \o "Vector asociativo), mejorados con la inclusión de prototipos (ver más adelante). Los nombres de las propiedades de los objetos son claves de tipo cadena: obj.x = 10 y obj['x'] = 10 son equivalentes, siendo [azúcar sintáctico](https://es.wikipedia.org/wiki/Az%C3%BAcar_sint%C3%A1ctico) la notación con punto. Las propiedades y sus valores pueden ser creados, cambiados o eliminados en tiempo de ejecución. La mayoría de propiedades de un objeto (y aquellas que son incluidas por la cadena de la herencia prototípica) pueden ser enumeradas a por medio de la instrucción de bucle for... in. JavaScript tiene un pequeño número de objetos predefinidos como son Function y Date.

**Evaluación en tiempo de ejecución**

JavaScript incluye la función eval que permite evaluar expresiones expresadas como cadenas en tiempo de ejecución. Por ello se recomienda que eval sea utilizado con precaución y que se opte por utilizar la función JSON.parse() en la medida de lo posible, pues resulta mucho más segura.

### Funcional

**Funciones de primera clase**

A las [funciones](https://es.wikipedia.org/wiki/Subrutinas) se les suele llamar ciudadanos de primera clase; son objetos en sí mismos. Como tal, poseen propiedades y métodos, como .call() y .bind().[27](https://es.wikipedia.org/wiki/JavaScript#cite_note-27)​ Una función anidada es una función definida dentro de otra. Esta es creada cada vez que la función externa es invocada. Además, cada función creada forma una [clausura](https://es.wikipedia.org/wiki/Clausura_(inform%C3%A1tica)); es el resultado de evaluar un ámbito conteniendo en una o más variables dependientes de otro ámbito externo, incluyendo constantes, variables locales y argumentos de la función externa llamante. El resultado de la evaluación de dicha clausura forma parte del estado interno de cada objeto función, incluso después de que la función exterior concluya su evaluación.[28](https://es.wikipedia.org/wiki/JavaScript#cite_note-FOOTNOTEFlanagan2006141-28)​

### Prototípico

**Prototipos**

JavaScript usa [prototipos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_basada_en_prototipos) en vez de [clases](https://es.wikipedia.org/wiki/Clase_(inform%C3%A1tica)) para el uso de [herencia](https://es.wikipedia.org/wiki/Herencia_(inform%C3%A1tica)).[29](https://es.wikipedia.org/wiki/JavaScript#cite_note-29)​ Es posible llegar a emular muchas de las características que proporcionan las clases en lenguajes orientados a objetos tradicionales por medio de prototipos en JavaScript.[30](https://es.wikipedia.org/wiki/JavaScript#cite_note-30)​

**Funciones como constructores de objetos**

Las funciones también se comportan como constructores. Prefijar una llamada a la función con la palabra clave new crear una nueva instancia de un prototipo, que heredan propiedades y métodos del constructor (incluidas las propiedades del prototipo de Object).[31](https://es.wikipedia.org/wiki/JavaScript#cite_note-31)​ ECMAScript 5 ofrece el método Object.create, permitiendo la creación explícita de una instancia sin tener que heredar automáticamente del prototipo de Object (en entornos antiguos puede aparecer el prototipo del objeto creado como null).[32](https://es.wikipedia.org/wiki/JavaScript#cite_note-32)​ La propiedad prototype del constructor determina el objeto usado para el prototipo interno de los nuevos objetos creados. Se pueden añadir nuevos métodos modificando el prototipo del objeto usado como constructor. Constructores predefinidos en JavaScript, como Array u Object, también tienen prototipos que pueden ser modificados. Aunque esto sea posible se considera una mala práctica modificar el prototipo de Object ya que la mayoría de los objetos en Javascript heredan los métodos y propiedades del objeto prototype, objetos los cuales pueden esperar que estos no hayan sido modificados.[33](https://es.wikipedia.org/wiki/JavaScript#cite_note-33)​

### Otras características

**Entorno de ejecución**

JavaScript normalmente depende del entorno en el que se ejecute (por ejemplo, en un [navegador web](https://es.wikipedia.org/wiki/Navegador_web)) para ofrecer objetos y métodos por los que los scripts pueden interactuar con el "mundo exterior". De hecho, depende del entorno para ser capaz de proporcionar la capacidad de incluir o importar scripts (por ejemplo, en [HTML](https://es.wikipedia.org/wiki/HTML) por medio del tag <script>). (Esto no es una característica del lenguaje, pero es común en la mayoría de las implementaciones de JavaScript.)

**Funciones variádicas**

Un número indefinido de parámetros pueden ser pasados a la función. La función puede acceder a ellos a través de los parámetros o también a través del objeto local arguments. Las [funciones variádicas](https://es.wikipedia.org/w/index.php?title=Funci%C3%B3n_vari%C3%A1dica&action=edit&redlink=1) también pueden ser creadas usando el método .apply().

**Funciones como métodos**

A diferencia de muchos lenguajes orientados a objetos, no hay distinción entre la definición de función y la definición de [método](https://es.wikipedia.org/wiki/M%C3%A9todo_(inform%C3%A1tica)). Más bien, la distinción se produce durante la llamada a la función; una función puede ser llamada como un método. Cuando una función es llamada como un método de un objeto, la palabra clave this, que es una variable local a la función, representa al objeto que invocó dicha función.

**Arrays y la definición literal de objetos**

Al igual que muchos lenguajes de script, arrays y objetos ([arrays asociativos](https://es.wikipedia.org/wiki/Vector_asociativo" \o "Vector asociativo) en otros idiomas) pueden ser creados con una sintaxis abreviada. De hecho, estos literales forman la base del formato de datos [JSON](https://es.wikipedia.org/wiki/JSON).

**Expresiones regulares**

JavaScript también es compatible con [expresiones regulares](https://es.wikipedia.org/wiki/Expresiones_regulares) de una manera similar a [Perl](https://es.wikipedia.org/wiki/Perl), que proporcionan una sintaxis concisa y poderosa para la manipulación de texto que es más sofisticado que las funciones incorporadas a los objetos de tipo string.[34](https://es.wikipedia.org/wiki/JavaScript#cite_note-34)​