El lenguaje de programación de Bitcoin denominado Script, **tiene su curva de aprendizaje como cualquier otro lenguaje**. Algunas características lo hacen algo más complicado y poco convencional de aprender, dado que es un lenguaje de muy bajo nivel y por ser Turing incompleto.

Conociendo ya cómo funciona técnicamente la Blockchain de Bitcoin, su arquitectura, los conceptos, teoremas, [bitcoin-cli](https://platzi.com/clases/2922-bitcoin-core-script/48689-conoce-bitcoin-cli/" \t "_blank), los contratos, entre otras partes, ya estás listo para dar tus primeros pasos en este lenguaje.

**Primeros pasos con Bitcoin Script**

Script es un lenguaje de programación que permite ejecutar funciones a través de una pila con ciertos datos y operandos denominados [OP\_CODES](https://github.com/bitcoin/bitcoin/blob/master/src/script/script.h), además de la utilización de las llaves públicas y privadas para tener los permisos necesarios sobre los BTC.

Tal vez te sorprendas cuando sepas que el *[white paper](https://bitcoin.org/bitcoin.pdf" \t "_blank)*[de Bitcoin](https://bitcoin.org/bitcoin.pdf" \t "_blank), no hace mención alguna a Bitcoin Script. Si bien Satoshi Nakamoto lo creó desde la primera versión de esta Blockchain en su código fuente desde el 2009.

El libro blanco de Bitcoin describe una transacción de la siguiente manera:

*Definimos una moneda electrónica como una cadena de firmas digitales.  
Cada dueño transfiere la moneda al próximo al firmar digitalmente un hash de la transacción previa y la clave pública del próximo.* (Bitcoin White Paper, página 2)

A pesar de sus limitaciones lógicas, Bitcoin Script **permite crear todo tipo de transacciones**. No sabemos si cuando Satoshi lo creó, lo hizo siendo consiente del potencial de esta herramienta que al día de hoy aún continúa creciendo y evolucionando. Tal vez lo sabía y era su intención que la comunidad explotara las características del lenguaje y le encontrara nuevos usos en el futuro.

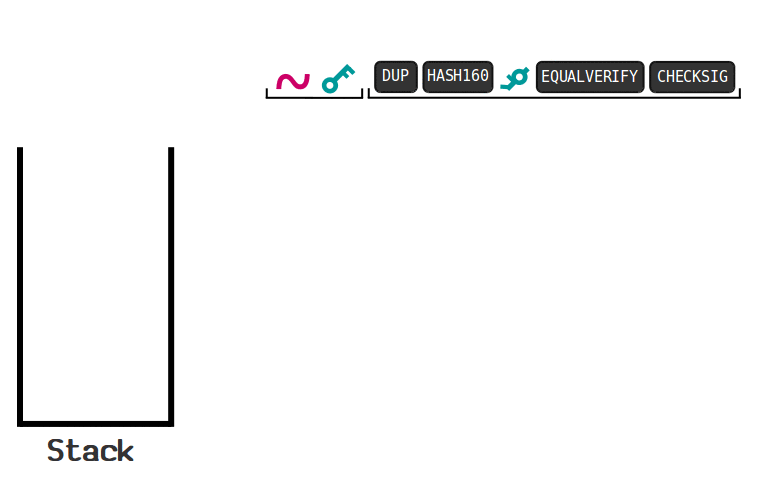
**¿Cómo funciona Bitcoin Script?**

Bitcoin Script permite bloquear salidas no gastadas ([UTXO](https://platzi.com/clases/2735-bitcoin-developers/46044-transacciones-outputs-e-inputs/)) con un script de bloqueo, y a través del cumplimiento de ciertas condiciones, permite gastarlas con otro script de desbloqueo. Siempre se ejecuta el script de desbloqueo primero que el script de bloqueo.

Todo lo que ocurre en la ejecución de un script, es booleano. Cada operación es verificada por si o por no para continuar con la ejecución del script en la pila de instrucciones.

Un Script se ejecuta siguiendo las siguientes premisas:

* Siempre de izquierda a derecha
* Los datos siempre se empujan a la pila
* OPCODES pueden sacar elementos, y opcional, empujar
* Válido si la cima y único elemento en la pila Ges 1 o más



Bitcoin Script es flexible. Si solo utilizáramos los OP\_CODES y las llaves públicas y privadas para validar las transacciones, su comportamiento sería mucho más limitado. Podemos crear todo tipo de bloqueos para las salidas y distintas combinaciones de los operadores y los datos que utilizamos.

**Veamos algunos ejemplos de Script:**

* Suma de dos números: [1 2 OPP\_ADD]
* Resta de dos números: [3 2 OPP\_ADD 4 OPP\_SUB]
* P2PKH (Script más común para crear una transacción): <signature> <pubKey> OP\_DUP OP\_HASH160 <pubKeyHash> OP\_EQUALVERIFY OP\_CHECKSIG

El lenguaje Bitcoin Script **ha ido solucionando bugs y evolucionado con los años**. Al día de hoy existen múltiples scripts estándar para crear transacciones, funciones especiales en el código fuente para validar los scripts como EvalScript, mejoras en la privacidad y ocultación de información, hasta llegar a complejos mecanismos como “Taproot” que hoy permiten desarrollar redes más poderosas como lo es [Lightning Network](https://platzi.com/clases/3141-bitcoin-lightning/49560-que-es-lighting-network/" \t "_blank).

**Conclusión**

Ahora entiendes cómo funciona por detrás Bitcoin Script. Se trata de un lenguaje de programación poco convencional y algo más difícil de utilizar, ya que los desarrolladores estamos acostumbrados a los condicionales, ciclos de iteración, clases, etc.

**RESUMEN CLASE 16: SCRIPT**

Definimos una moneda electrónica como una cadena de firmas digitales. Cada dueño transfiere la moneda al próximo al firmar digitalmente un hash de la transacción previa y la clave pública del próximo.

I.- SCRIPT

* No es nombrado en el whitepaper.
* EvalScript (funcion).
* Máquina de pila que evalúa y retorna validez.
* Soporte para todo tipo de transacción.
* Cada nodo solo necesita entender la transacción para evaluarla.
* Predicado (Teorema de post).

II.- SCRIPT HISTORIA

* Scriptsig
* EvalScript
* P2SH
* Segregated Witness
* Taproot (Firmar)

III.- QUE ES SCRIPT ?

* Lenguaje de programación usado como mecanismo que bloquea salidas.
* Un script de bloqueo se agrega en cada salida.
* Un script de desbloqueo debe proveerse para desbloquear una salida.

IV.- COMPONENTES

* Datos (Llaves públicas y firmas).
* OPCODES: funciones que permiten operar los datos.

CLASE16 Imagen1_Script.png

V.- COMO ES EJECUTADO

* Siempre de izquierda a derecha.
* Los datos siempre se empujan a la pila
* OPCODES pueden sacar elementos, y opcional, empujar.
* Válido si la cima y único elemento en la pila Ges 1 o más.

CLASE16 Imagen2_Script.png

VI.- DONDE LO ENCUENTRAS ?

* Script que bloquea en cada salida que se crea en una transacción.
* Script que desbloquea cada vez que quieres usar una salida como entrada en una nueva transacción.
* Siempre se ejecuta el script de desbloqueo primero que el script de bloqueo.

VII.- POR QUE LO USAMOS ?

* ¿ Por qué no usar una llave pública y la firma solamente y no usar OPCODES ?
* Podemos crear distintos tipos de bloqueos con distintas combinaciones, flexibilidad.

EJEMPLOS

* Matemático
  + Suma: ‘[1 2 OPP\_ADD]’
  + Resta: ‘[3 2 OPP\_ADD 4 OPP\_SUB]’
* P2PKH
  + <signature> <pubKey> OP\_DUP OP\_HASH160 <pubKeyHash> OP\_EQUALVERIFY OP\_CHECKSIG

VIII.- SCRIPT ESTANDAR

* P2PK
* P2PKH
* P2MS
* P2SH
* NULL