

DICIONÁRIO

ROBLOX + LUA

Aprenda o significado dos principais comandos
e serviços usados no Roblox Studio!

DICIONÁRIO ROBLOX + LUA — PROGRAMAÇÃO FOR KIDS (VERSÃO 2025)

👋 BEM-VINDO AO DICIONÁRIO ROBLOX + LUA!

Este material foi criado especialmente para você que está aprendendo a programar e quer entender os principais termos usados no Roblox Studio. Aqui você vai encontrar explicações curtas, simples e diretas sobre os comandos, serviços e conceitos que aparecem quando criamos jogos.

Explore, aprenda e divirta-se programando!

— Equipe Programação for Kids 💻

ÍNDICE

🟦 SINTAXE LUA
🟩 SERVIÇOS ROBLOX
🟥 OBJETOS E PROPRIEDADES
🟪 EVENTOS E REDE
🟨 INTERFACE (GUI)
🟧 ECONOMIA E DADOS
⚪ FÍSICA E MOVIMENTO
💡 EXTRAS

🟦 SINTAXE LUA

- local → Cria uma variável que só existe dentro do script atual.
- function → Define uma função, um bloco de código reutilizável.
- if / then / else → Estrutura condicional: executa algo se uma condição for verdadeira.
- for / while / repeat → Estruturas de repetição (loops).
- return → Retorna um valor de uma função.
- true / false / nil → Valores lógicos e nulo.
- and / or / not → Operadores lógicos usados em condições.
- == / ~= / < / > / <= / >= → Operadores de comparação.
- .. → Junta (concatena) textos.
- print() → Mostra mensagens no console (para depurar scripts).
- math.random() → Gera números aleatórios.
- string.sub() → Pega parte de um texto.
- task.wait() → Faz o script pausar por alguns segundos.
- pairs / ipairs → Iteram sobre listas e tabelas.
- table.insert() / table.remove() → Adiciona ou remove itens de uma tabela.

🟩 SERVIÇOS ROBLOX

- game:GetService() → Comando usado para acessar qualquer serviço do Roblox.
- Players → Guarda todos os jogadores conectados ao jogo.
- Workspace → Contém todos os objetos visíveis dentro do jogo.
- ServerStorage → Guarda objetos que só o servidor pode acessar.
- ReplicatedStorage → Guarda objetos visíveis por cliente e servidor.
- ReplicatedFirst → Carrega antes de o jogo começar, ideal para telas de loading.
- StarterGui → Armazena interfaces (menus, botões, textos).
- StarterPlayer → Armazena scripts e configurações do jogador.
- Lighting → Controla luz, sombras e ambiente do jogo.
- TweenService → Cria animações suaves (movimento, cor, transparência).
- DataStoreService → Salva e carrega informações do jogador.
- TeleportService → Move jogadores entre diferentes jogos ou mapas.

🟩 SERVIÇOS ROBLOX

- MarketplaceService → Gerencia compras e transações com Robux.
- RunService → Detecta se o script está rodando no servidor ou no cliente.
- UserInputService → Detecta teclas, mouse e toques do jogador.
- ContextActionService → Liga ações a comandos, teclas ou botões.
- HttpService → Permite que o jogo envie e receba dados da web.
- MessagingService → Envia mensagens entre servidores diferentes.
- BadgeService → Entrega medalhas e conquistas aos jogadores.
- Teams → Define grupos de jogadores dentro do jogo.

🟥 OBJETOS E PROPRIEDADES

- Part → Um bloco físico no jogo (como chão, parede, plataforma).
- Model → Agrupa várias partes e objetos em um conjunto.
- Folder → Serve para organizar objetos dentro do jogo.
- Humanoid → Dá vida, saúde e animação a personagens.
- HumanoidRootPart → A parte principal que define a posição do personagem.
- Tool → Objeto que o jogador pode equipar (espada, arma, item).
- Accessory → Itens de visual (chapéus, mochilas, etc.).
- CFrame → Define posição e rotação de um objeto em 3D.
- Vector3 → Representa coordenadas (x, y, z).
- Color3 → Representa cores RGB.
- Anchored → Faz o objeto ficar fixo, sem cair.
- CanCollide → Define se o objeto colide ou atravessa outros.
- Transparency → Define o quanto o objeto é visível.
- Size → Define o tamanho do objeto.
- Orientation → Define o ângulo de rotação do objeto.
- Velocity → Define a velocidade de movimento.

🟥 OBJETOS E PROPRIEDADES

- Parent → Mostra onde o objeto está localizado dentro da hierarquia.
- Clone() → Cria uma cópia idêntica do objeto.
- Destroy() → Remove o objeto do jogo.
- WaitForChild() → Espera um objeto aparecer antes de usá-lo.
- FindFirstChild() → Procura um objeto dentro de outro.

🟪 EVENTOS E REDE

- RemoteEvent → Canal para enviar mensagens entre Cliente e Servidor sem retorno.
- RemoteFunction → Comunicação Cliente ↔️ Servidor com resposta (pergunta e resposta).
- FireServer → Envia dados do Cliente para o Servidor usando um RemoteEvent.
- FireClient → Envia dados do Servidor para um Cliente específico (RemoteEvent).
- OnServerEvent / OnClientEvent → Ouvem mensagens recebidas do outro lado (evento).
- InvokeServer / InvokeClient → Chamadas que esperam resposta (RemoteFunction).
- Replicação → Processo de sincronizar mudanças entre Servidor e Cliente.
- Cliente (Client) → Parte do jogo que roda no dispositivo do jogador (UI/efeitos).
- Servidor (Server) → Parte autoritativa que valida regras e salva dados.
- Latência (Lag) → Atraso entre enviar e receber dados pela rede.
- Debounce → Técnica para evitar que um evento dispare muitas vezes seguidas.

🟪 EVENTOS E REDE

- ProximityPrompt → Interação que aparece quando o jogador se aproxima de algo.
- ClickDetector → Detecta cliques do mouse em um objeto 3D.
- BindableEvent → Mensagem local entre scripts no mesmo lado (sem rede).
- BindableFunction → Função local entre scripts que retorna um valor.
- Touch/Touched → Evento disparado quando algo encosta em uma parte.
- GetPropertyChangedSignal → Evento disparado quando uma propriedade muda.

🟨 INTERFACE (GUI)

- ScreenGui → Contêiner principal de interface, aparece na tela do jogador.
- Frame → Caixa de interface que pode agrupar outros elementos.
- TextLabel → Exibe um texto fixo na tela.
- TextButton → Botão de texto que o jogador pode clicar.
- ImageLabel → Mostra uma imagem estática na interface.
- ImageButton → Botão de imagem clicável.
- ViewportFrame → Mostra objetos 3D dentro da interface 2D.
- UIListLayout → Organiza elementos em uma lista vertical ou horizontal.
- UIGridLayout → Organiza elementos em forma de grade.
- UIPadding → Adiciona espaçamento interno em um elemento.
- UICorner → Deixa cantos arredondados.
- UIStroke → Adiciona contorno (borda) a um elemento.
- UIScale / UITextSizeConstraint → Controlam escala e tamanho de texto/GUI.
- UDim2 (Position/Size) → Define posição e tamanho proporcionais na tela.
- AnchorPoint → Define o ponto de ancoragem para posicionamento.

🟨 INTERFACE (GUI)

- ZIndex → Controla o que fica por cima ou por baixo na interface.
- Visible → Mostra ou esconde um elemento da interface.
- TextScaled / RichText → Ajusta tamanho automaticamente / permite formatação.
- MouseButton1Click / Activated → Eventos comuns de clique em botões.

🟧 ECONOMIA E DADOS

- leaderstats → Pasta especial no jogador para mostrar estatísticas na lista de jogadores.
- IntValue / NumberValue / StringValue / BoolValue → Armazenam valores simples no jogo.
- DataStore → Espaço para salvar dados do jogador entre sessões.
- GetAsync / SetAsync → Ler e escrever dados simples no DataStore.
- UpdateAsync → Atualiza dados com segurança evitando sobrescrever.
- OrderedDataStore → DataStore ordenado para rankings (leaderboards).
- Autosave → Salvamento automático de tempos em tempos.
- Backup / Recuperação → Estratégia para evitar perda de dados em falhas.
- Throttle / Limites → Restrições de frequência ao usar DataStores.
- Coins / Moedas → Exemplo comum de economia básica.
- XP / Nível → Sistema de progressão do jogador.
- Game Pass / Developer Product → Tipos de compras com Robux (única vs. repetível).

🟧 ECONOMIA E DADOS

- Transações seguras → Confirmar compra antes de dar o item (checagem no servidor).
- Logs de economia → Registrar mudanças importantes (auditoria simples).

FÍSICA E MOVIMENTO

- Raycast → Linha invisível usada para detectar colisões em um caminho.
- Gravity → Força que puxa objetos para baixo no mundo.
- Anchored / CanCollide / CanTouch / CanQuery → Propriedades que controlam física.
- Massless → Faz uma peça não contribuir com massa do conjunto.
- AssemblyLinearVelocity / AssemblyAngularVelocity → Velocidades do conjunto físico.
- BodyGyro / BodyPosition (legado) → Objetos antigos de controle físico (substituídos).
- LinearVelocity / AngularVelocity → Controlam velocidade linear/ang. modernas.
- AlignPosition / AlignOrientation → Mantêm objetos alinhados a posições/rotações.
- HingeConstraint / SpringConstraint → Juntas de dobradiça e mola.
- PrismaticConstraint → Movimento restrito a um eixo (tipo trilho).
- CustomPhysicalProperties → Define densidade, atrito e elasticidade.

⚪ Física e Movimento

- WalkSpeed / JumpPower → Velocidade e pulo do personagem (Humanoid).
- MoveDirection → Direção atual de movimento do personagem.
- PlatformStand → Deixa o personagem imóvel como se estivesse numa plataforma.

💡 EXTRAS

- Script / LocalScript / ModuleScript → Tipos de scripts (servidor, cliente, módulo).
- require() → Importa um ModuleScript para reutilizar código.
- Attributes → Propriedades personalizadas que você pode criar em objetos.
- CollectionService (Tags) → Adiciona etiquetas para organizar e buscar objetos.
- PathfindingService → Calcula caminhos para NPCs desviarem de obstáculos.
- Camera / CurrentCamera → Controla a câmera vista pelo jogador.
- CameraType → Modos de câmera (seguindo, fixa, script etc.).
- StarterCharacterScripts / StarterPlayerScripts → Onde colocar scripts padrão do jogador.



“Aprender a programar é mais do que dominar códigos — é despertar o poder de criar o futuro! 🌟
Continue sua jornada e transforme ideias em jogos incríveis no Roblox Studio.”
👉 programacaoforkids.com.br/cursos/curso-roblox-lua