

Oracle® Database

JavaScript Developer's Guide



Release 23ai
F56885-12
April 2025

ORACLE®

Contributors: M. Bach, L. Braun-Lohrer, H. Kasture, A. Ulrich, G. Venzl, M. Brantner, L. Daynes, H. Guiroux, A. Schubert, A. Burlison, M. Keppner, A. Kashuba, N. Sheikh, A.A. Baha, D. Adams

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

1 Changes in This Release for JavaScript Developer's Guide

July 2024, Release Update 23.5	1-1
January 2025, Release Update 23.7	1-1
April 2025, Release Update 23.8	1-2

2 Introduction to Oracle Database Multilingual Engine for JavaScript

The Need for a Multilingual Engine	2-2
Overview of JavaScript	2-2
Overview of Multilingual Engine for JavaScript	2-3
JavaScript Implementation Details	2-4
Invoking JavaScript in the Database	2-5
Introduction to Dynamic Execution	2-5
Introduction to MLE Module Calls	2-6
About MLE Execution Contexts	2-8
About Restricted Execution Contexts	2-9
Introduction to Debugging JavaScript Code	2-10

3 MLE JavaScript Modules and Environments

Using JavaScript Modules in MLE	3-1
Managing JavaScript Modules in the Database	3-3
Naming JavaScript Modules	3-3
Creating JavaScript Modules in the Database	3-4
Storing JavaScript Code in Databases Using Single-Byte Character Sets	3-5
Code Analysis	3-5
Preparing JavaScript code for MLE Module Calls	3-6
Additional Options for Providing JavaScript Code to MLE	3-8
Specifying Module Version Information and Providing JSON Metadata	3-9
Drop JavaScript Modules	3-10
Alter JavaScript Modules	3-10
Overview of Built-in JavaScript Modules	3-11
Dictionary Views Related to MLE JavaScript Modules	3-11
USER_SOURCE	3-12

USER_MLE_MODULES	3-13
Specifying Environments for MLE Modules	3-13
Creating MLE Environments in the Database	3-14
Naming MLE Environments	3-14
Creating an Empty MLE Environment	3-15
Creating an Environment as a Clone of an Existing Environment	3-16
Using MLE Environments for Import Resolution	3-16
Providing Language Options	3-18
Dropping MLE Environments	3-19
Modifying MLE Environments	3-19
Altering Language Options	3-20
Modifying Module Imports	3-20
Dictionary Views Related to MLE JavaScript Environments	3-20
USER_MLE_ENVS	3-21
USER_MLE_ENV_IMPORTS	3-21

4 Overview of Dynamic MLE Execution

About Dynamic JavaScript Execution	4-1
Dynamic Execution Workflow	4-2
Providing JavaScript Code Inline	4-2
Loading JavaScript Code from Files	4-3
Returning the Result of the Last Execution	4-6

5 Overview of Importing MLE JavaScript Modules

JavaScript Module Hierarchies	5-2
Resolving Import Names Using MLE Environments	5-2
Export Functionality	5-3
Named Exports	5-3
Default Exports	5-4
Private Identifiers	5-5
Import Functionality	5-5
Module Objects	5-5
Named Imports	5-6
Default Imports	5-7

6 MLE JavaScript Functions

Call Specifications for Functions	6-1
Creating a Call Specification for an MLE Module	6-1
Components of an MLE Call Specification	6-4

MLE Module Clause	6-5
ENV Clause	6-5
SIGNATURE Clause	6-5
Creating an Inline MLE Call Specification	6-7
Components of an Inline MLE Call Specification	6-10
Accessing Built-in Modules Using JavaScript Global Variables	6-11
Choosing Inline Versus Module MLE Call Specifications	6-12
Runtime Isolation for an MLE Call Specification	6-12
Dictionary Views for Call Specifications	6-15
OUT and IN OUT Parameters	6-16

7 Calling PL/SQL and SQL from the MLE JavaScript SQL Driver

Introduction to the MLE JavaScript SQL Driver	7-1
Working with the MLE JavaScript Driver	7-2
Connection Management in the MLE JavaScript Driver	7-3
Introduction to Executing SQL Statements	7-3
Processing Comparison Between node-oracledb and mle-js-oracledb	7-6
Selecting Data Using the MLE JavaScript Driver	7-6
Direct Fetch: Arrays	7-7
Direct Fetch: Objects	7-8
Fetching Rows as ResultSets: Arrays	7-9
Fetching Rows as ResultSets: Iterating Over ResultSet Objects	7-10
Data Modification	7-11
Bind Variables	7-11
Using Bind-by-Name vs Bind-by-Position	7-12
Named Bind Variables	7-12
Positional Bind Variables	7-14
RETURNING INTO Clause	7-15
Batch Operations	7-16
PL/SQL Invocation from the MLE JavaScript SQL Driver	7-18
Error Handling in SQL Statements	7-20
Working with JSON Data	7-25
Using Large Objects (LOB) with MLE	7-30
Writing LOBs	7-30
Reading LOBs	7-31
API Differences Between node-oracledb and mle-js-oracledb	7-32
Synchronous API and Error Handling	7-32
Connection Handling	7-33
Transaction Management	7-34
Type Mapping	7-34
Unsupported Data Types	7-37

Miscellaneous Features Not Available with the MLE JavaScript SQL Driver	7-37
Introduction to the PL/SQL Foreign Function Interface	7-38
Object Resolution Using FFI	7-39
Provide Arguments to a Subprogram Using FFI	7-42

8 Working with SODA Collections in MLE JavaScript Code

High-Level Introduction to Working with SODA for In-Database JavaScript	8-2
SODA Objects	8-3
Using SODA for In-Database JavaScript	8-4
Getting Started with SODA for In-Database JavaScript	8-6
Creating a Document Collection with SODA for In-Database JavaScript	8-8
Opening an Existing Document Collection with SODA for In-Database JavaScript	8-9
Checking Whether a Given Collection Exists with SODA for In-Database JavaScript	8-9
Discovering Existing Collections with SODA for In-Database JavaScript	8-10
Dropping a Document Collection with SODA for In-Database JavaScript	8-11
Creating Documents with SODA for In-Database JavaScript	8-12
Inserting Documents into Collections with SODA for In-Database JavaScript	8-14
Saving Documents into Collections with SODA for In-Database JavaScript	8-15
SODA for In-Database JavaScript Read and Write Operations	8-15
Finding Documents in Collections with SODA for In-Database JavaScript	8-17
Replacing Documents in a Collection with SODA for In-Database JavaScript	8-22
Removing Documents from a Collection with SODA for In-Database JavaScript	8-24
Indexing the Documents in a Collection with SODA for In-Database JavaScript	8-25
Getting a Data Guide for a Collection with SODA for In-Database JavaScript	8-27
Handling Transactions with SODA for In-Database JavaScript	8-29
Creating Call Specifications Involving the SODA API	8-29

9 Post-Execution Debugging of MLE JavaScript Modules

Specifying Debugpoints	9-2
Debugpoint Locations	9-2
Debugpoint Actions	9-2
Debugpoint Conditions	9-4
Managing Debugpoints	9-4
Debugging Security Considerations	9-6
COLLECT DEBUG INFO Privilege for MLE Modules	9-6
Analyzing Debug Output	9-7
Textual Representation of Debug Output	9-7
Analyzing Debug Output Using Developer Tools	9-10
Error Handling in MLE	9-10
Errors in Callouts	9-13

Accessing stdout and stderr from JavaScript	9-13
Accessing stdout and stderr for MLE Modules	9-13
Accessing stdout and stderr for Dynamic MLE	9-15

10 MLE Security

System and Object Privileges Required for Working with JavaScript in MLE	10-1
Necessary Privileges for the Execution of JavaScript Code	10-2
Necessary Privileges for Using the NoSQL API	10-2
Necessary Privileges for Creating MLE Schema Objects	10-3
Necessary Privileges for Creating MLE Modules and Environments in ANY Schema	10-3
Necessary Privileges for Post-Execution Debugging	10-4
Security Considerations for MLE	10-4
MLE_PROG_LANGUAGES Initialization Parameter	10-5
Execution Contexts	10-5
Runtime State Isolation	10-6
Database Security Model	10-8
Considerations for Using MLE Call Specifications and Modules from Different Schemas	10-9
Auditing MLE Operations in Oracle Database	10-10
JavaScript Security Best Practices	10-10
Using Bind Variables for Security and Performance	10-10
Generic Database and PL/SQL Specific Security Considerations	10-12
Supply Chain Security	10-13
Software Bill of Material	10-14
Using the Database to Store State	10-14
Disabling Multilingual Runtime	10-16
MLE Security Examples	10-16
Business Logic Stored in MLE Modules	10-16
Generic Data Processing Libraries	10-18
Generic Libraries in Business Logic	10-19

A MLE Type Conversions

MLE JavaScript Support for JSON	A-4
MLE JavaScript Support for the VECTOR Data Type	A-6

Index

List of Examples

3-1	Creating a JavaScript Module in the Database	3-4
3-2	Create a Call Specification for a Public Function	3-6
3-3	Public and Private Functions in a JavaScript Module	3-7
3-4	Providing JavaScript Source Code Using a BFILE	3-8
3-5	Providing JavaScript Source Code Using a CLOB	3-8
3-6	Specification of a VERSION string in CREATE MLE MODULE	3-9
3-7	Addition of JSON Metadata to the MLE Module	3-10
3-8	Drop an MLE Module	3-10
3-9	Drop an MLE Module Using IF EXISTS	3-10
3-10	Alter an MLE Module	3-10
3-11	Externalize JavaScript Module Source Code	3-12
3-12	Find MLE Modules Defined in a Schema	3-13
3-13	Map Identifier to JavaScript Module	3-16
3-14	Import Module Functionality	3-17
3-15	List Available MLE Environments Using USER_MLE_ENVS	3-21
3-16	List Module Import Information Using USER_MLE_ENV_IMPORTS	3-21
4-1	Using the Q-Quote Operator to Provide JavaScript Code Inline with PL/SQL	4-2
4-2	Loading JavaScript code from a BFILE with DBMS_LOB.LOADCLOBFROMFILE()	4-3
4-3	Loading JavaScript Code from a BFILE by Referencing an MLE Module from DBMS_MLE	4-5
4-4	Returning the Result of the Last Execution	4-6
5-1	Use an MLE Environment to Map an Import Name to a Module	5-2
5-2	Function Export using Named Exports	5-3
5-3	Function Export Using Export Keyword Inline	5-4
5-4	Export a Class Using a Default Export	5-4
5-5	Named Export of Single Function	5-5
5-6	Module Object Definition	5-6
5-7	Named Imports Using Specified Identifiers	5-6
5-8	Named Imports with Aliases	5-7
5-9	Default Import	5-7
5-10	Default Import with Built-in Module	5-7
6-1	Creating MLE Call Specifications	6-2
6-2	Simple Inline MLE Call Specification	6-8
6-3	Inline MLE Call Specification Returning JSON	6-9
6-4	Execution Context Dependencies	6-13
6-5	Show JavaScript Call Specification Metadata	6-15
6-6	OUT and IN OUT Parameters with JavaScript	6-16

7-1	Getting Started with the MLE JavaScript SQL Driver	7-3
7-2	Use Global Variables to Simplify SQL Execution	7-5
7-3	Selecting Data Using Direct Fetch: Arrays	7-7
7-4	Selecting Data Using Direct Fetch: Objects	7-8
7-5	Fetching Rows Using a ResultSet	7-9
7-6	Using the Iterable Protocol with ResultSets	7-10
7-7	Updating a Row Using the MLE JavaScript SQL Driver	7-11
7-8	Using Named Bind Variables	7-13
7-9	Using Positional Bind Variables	7-14
7-10	Using the RETURNING INTO Clause	7-15
7-11	Performing a Batch Operation	7-17
7-12	Calling PL/SQL from JavaScript	7-18
7-13	SQL Error Handling Inside a JavaScript Function	7-21
7-14	Error Handling Using JavaScript throw() Command	7-23
7-15	Inserting JSON Data into a Database Table	7-25
7-16	Use JavaScript to Manipulate JSON Data	7-28
7-17	Inserting a CLOB into a Table	7-30
7-18	Read an LOB	7-31
7-19	Using JavaScript Native Data Types vs Using Wrapper Types	7-35
7-20	Overriding the Global oracledb.fetchAsPlsqlWrapper Property	7-36
8-1	SODA with MLE JavaScript General Workflow	8-6
8-2	Opening an Existing Document Collection	8-9
8-3	Fetching All Existing Collection Names	8-10
8-4	Filtering the List of Returned Collections	8-10
8-5	Dropping a Collection	8-11
8-6	Creating SODA Documents	8-13
8-7	Inserting a SODA Document into a Collection	8-14
8-8	Inserting an Array of Documents into a Collection	8-15
8-9	Finding a Document by Key	8-17
8-10	Looking up Documents Using Multiple Keys	8-18
8-11	Using a QBE to Filter Documents in a Collection	8-19
8-12	Using skip() and limit() in a Pagination Query	8-20
8-13	Specifying Document Versions	8-21
8-14	Counting the Number of Documents Found	8-21
8-15	Replacing a Document in a Collection and Returning the Result Document	8-23
8-16	Removing a Document from a Collection Using a Document Key	8-24
8-17	Removing JSON Documents from a Collection Using a Filter	8-24

8-18	Creating a B-Tree Index for a JSON Field with SODA for In-Database JavaScript	8-25
8-19	Creating a JSON Search Index with SODA for In-Database JavaScript	8-26
8-20	Dropping an Index with SODA for In-Database JavaScript	8-27
8-21	Generating a Data Guide for a Collection	8-27
8-22	Use SODA for In-Database JavaScript	8-29
9-1	JSON Template for Specifying Debugpoints	9-2
9-2	JSON Template for Specifying Watch Action	9-3
9-3	JSON Template for Specifying Snapshot Action	9-3
9-4	Watching a Variable in an MLE Module	9-4
9-5	Enabling Debugging of an MLE Module	9-5
9-6	Obtain Textual Representation of Debug Output	9-7
9-7	Throwing ORA-04161 Error and Querying the Stack Trace	9-11
9-8	Redirect stdout to CLOB and DBMS_OUTPUT for MLE Module	9-14
9-9	Redirect stdout to CLOB and DBMS_OUTPUT for Dynamic MLE	9-15
10-1	Runtime State Isolation Scenario	10-6
10-2	Using Bind Variables Rather than String Concatenation	10-11
10-3	Use DBMS_ASSERT to Verify Valid Input	10-11
10-4	Using Bind Variables Rather than String Concatenation	10-15
10-5	Use DBMS_ASSERT to Verify Valid Input	10-15
10-6	Business Logic Stored in MLE Modules	10-17
10-7	Generic Data Processing Libraries	10-18
10-8	Use Generic Libraries in Business Logic	10-19
A-1	Use VECTOR Data Type with MLE	A-7

List of Figures

6-1	MLE Call Specification Syntax	6-4
6-2	signature_clause ::=	6-6
6-3	path_spec ::=	6-6
6-4	import_spec ::=	6-6
6-5	MLE Inline Call Specification Syntax	6-10
8-1	SODA for In-Database JavaScript Basic Workflow	8-2
8-2	SODA for In-Database JavaScript Simplified Workflow	8-3

List of Tables

3-1	JavaScript Language Options	3-19
6-1	Components of an MLE Call Specification	6-4
6-2	Components of an Inline MLE Call Specification	6-10
8-1	Overview of Nonterminal Methods for Read Operations	8-16
8-2	Overview of Terminal Methods for Read Operations	8-16
8-3	Overview of Terminal Methods for Write Operations	8-16
A-1	Supported Mappings from SQL and PL/SQL Types to JavaScript Types	A-2
A-2	Supported Mappings from JavaScript Types to SQL Types	A-3
A-3	Mapping from JSON Attribute Types and Values to JavaScript Types and Values	A-5
A-4	Mapping from JavaScript Types and Values to JSON Attributes and Values	A-5
A-5	Mapping from VECTOR Data Type to JavaScript Types	A-6
A-6	Mapping from JavaScript Types to VECTOR Data Type	A-6