MS SQL & JSON

Damir Matešić, M. Sc. Inf.

Damir Matešić, M. Sc. Inf.

- Senior Database Developer @SPAN.eu
- Microsoft Data Platform MVP

- Leading PASS MS SQL UG Coratia
- Introduced SQL Saturday in Croatia
- Co-founder & organizer of #Dataweekender…

- W: blog.matesic.info
- @: damir@matesic.info
- in: linkedin.com/in/dmatesic

Agenda

1. JSON
2. SQL 2 JSON
3. JSON 2 SQL
4. Modifying JSON data
5. T&T

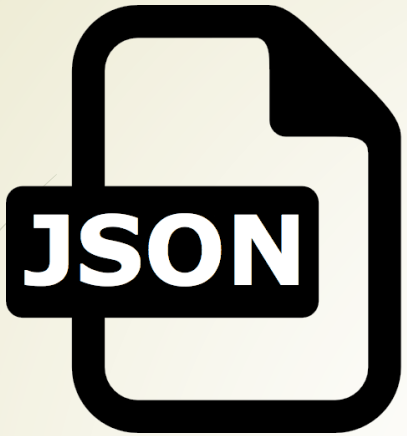
JSON 1/2

- **J**avaScript **O**bject **N**otation
- language in depended
- open standard format
- simple and very popular
- JSON objects are human readable lists of key-value pairs.

```
{
    "Name": "John Doe",
    "BlogURL": "http://blog.matesic.info",
    "Born": 1979,
    "Spouse": null,
    "BornAfterWoodstock": true,
    "FavoriteDrinks": [
        {
            "Name": "Gin and tonic",
            "Drink": "Occasionally"
        },
        {
            "Name": "Craft beer",
            "Drink": "Occasionally"
        },
        {
            "Name": "Coffe with milk",
            "Drink": "Daily"
        },
        {
            "Name": "Cold water",
            "Drink": "Daily"
        }
    ],
    "Parents": {
        "Mom": "Iva",
        "Dad": "Boris"
    }
}
```

JSON 2/2

- Supported data types:
  - **String** - escaped Unicode text surrounded by double quotes
  - **Number** - double-precision float
  - **Boolean** - true/false written in lowercase
  - **null** - represents a null value
- Escaping rules
  - Quotation mark (") -> \"
  - Reverse solidus (\) -> \\
  - Solidus (/) -> \/
  - Backspace -> \b
  - Form feed -> \f
  - New line -> \n
  - Carriage return -> \r
  - Horizontal tab -> \t
  - Control characters (0-31) -> **\u**<code> (e.g. CHAR(0) -> **\u**0000)



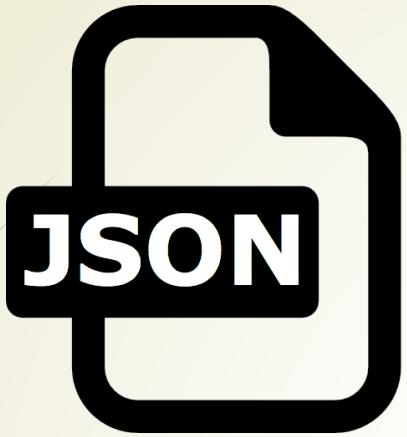
VS


```
[
  {
    "CustomerID": 1,
    "CustomerName": "Tailspin Toys (Head Office)",
    "PhoneNumber": "(308) 555-0100",
    "FaxNumber": "(308) 555-0101",
    "WebsiteURL": "http:\/\/www.tailspintoys.com",
    "DataDateTime": "2018-10-05T16:06:36.200"
  }
]
```

```
<Customer>
  <CustomerID>1</CustomerID>
  <CustomerName>Tailspin Toys (Head Office)</CustomerName>
  <PhoneNumber>(308) 555-0100</PhoneNumber>
  <FaxNumber>(308) 555-0101</FaxNumber>
  <WebsiteURL>http://www.tailspintoys.com</WebsiteURL>
  <DataDateTime>2018-10-05T16:07:52.813</DataDateTime>
</Customer>
```

Results | Messages

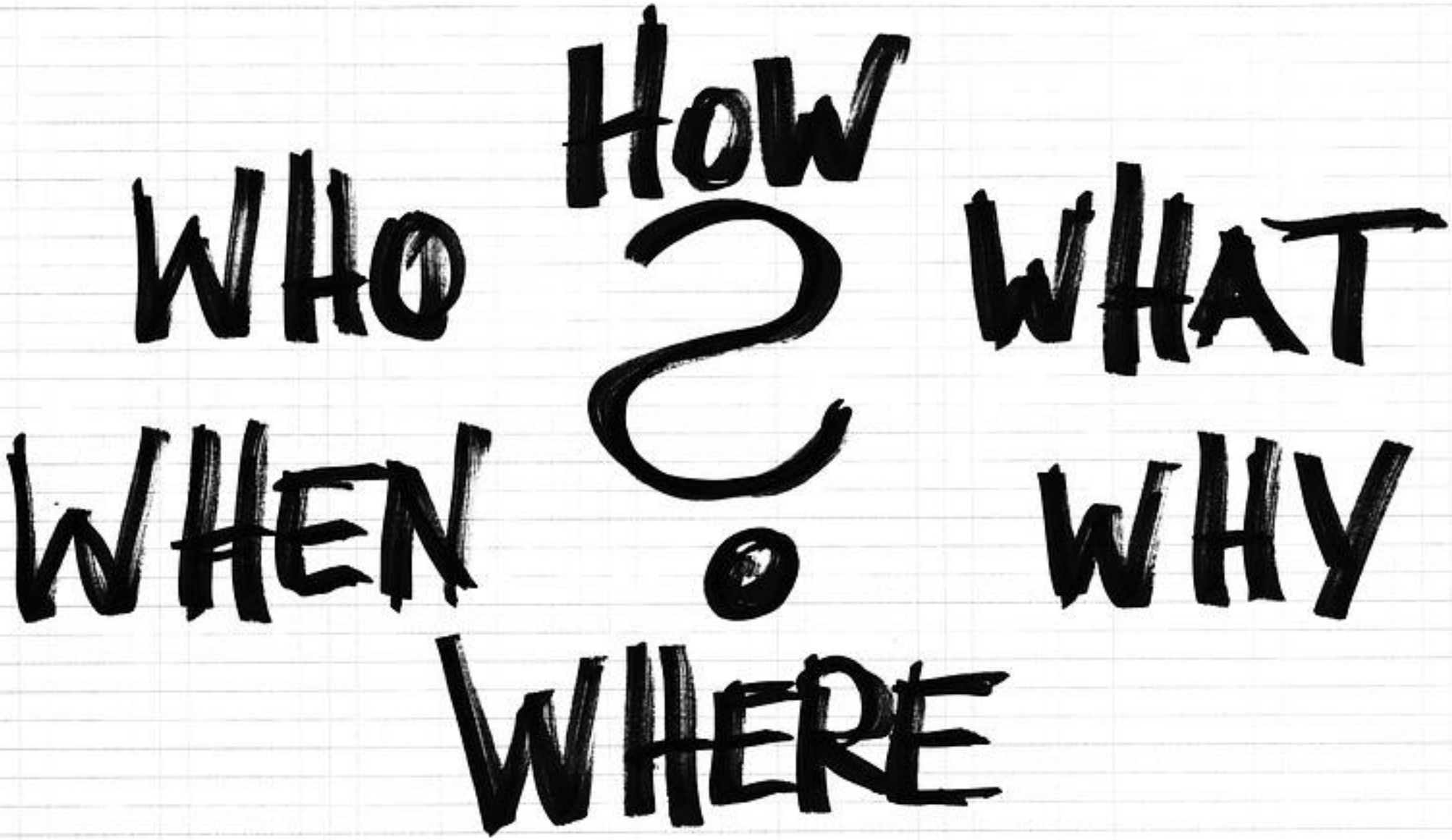
	CustomerID	CustomerName	PhoneNumber	FaxNumber	WebsiteURL	DataDateTime
1	1	Tailspin Toys (Head Office)	(308) 555-0100	(308) 555-0101	http://www.tailspintoys.com	2020-05-21 11:24:08.990



VS

- arrays and objects
- can store only data
- less verbose and easier to read
- less data
- SQL:
  - NVARCHAR -> COMPRESS ?!?!
  - index problem

- tree structure
- can store more complex data types
- can store additional information's
- more robust
- SQL:
  - native XML data type

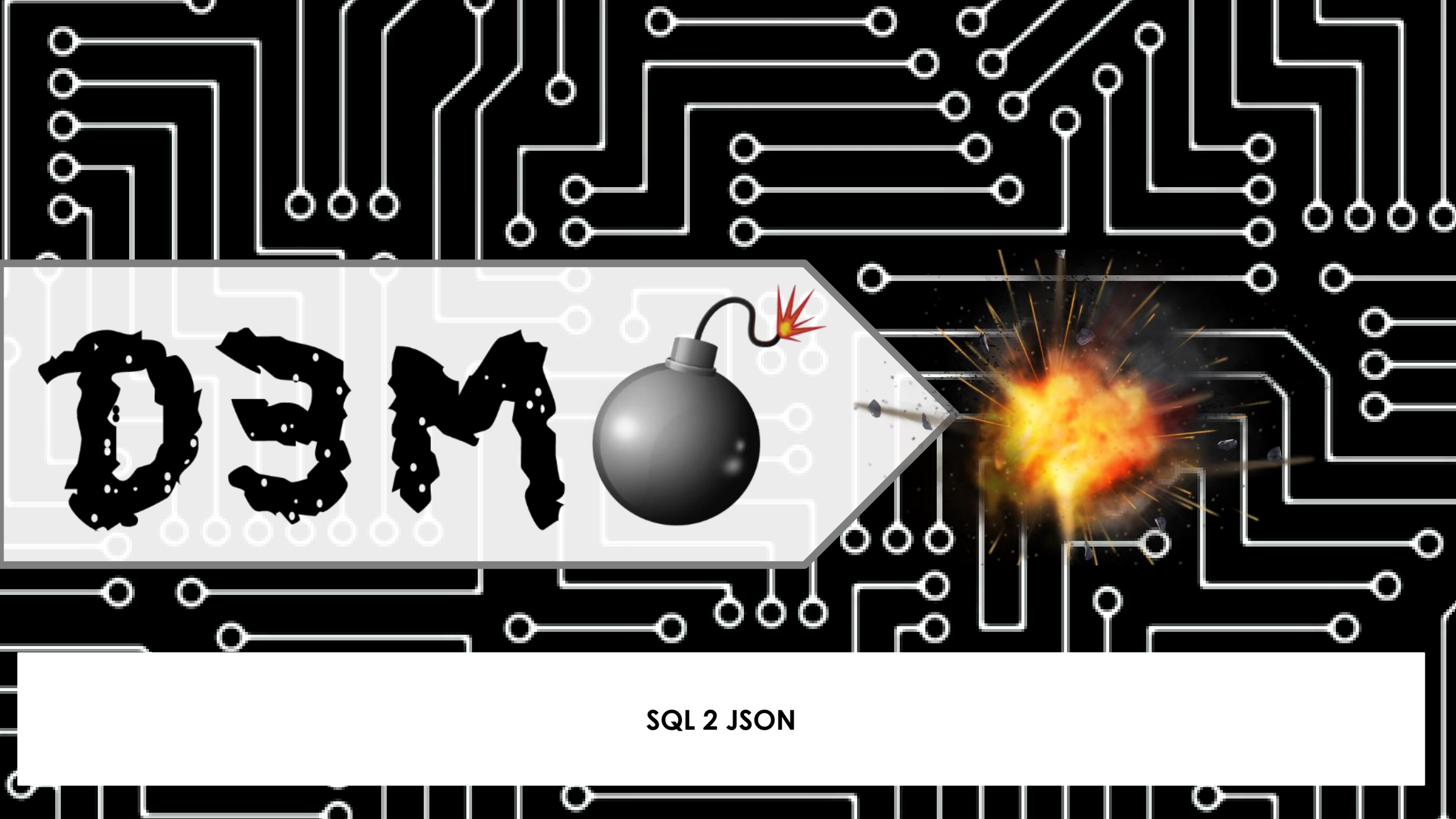

HOW
WHO
WHAT
WHEN
?
WHY
WHERE

SQL 2 JSON

- Pretty much like creating XML data (FOR XML) -> **FOR JSON**
- Two modes supported:
  - **FOR JSON AUTO** – output will be formatted automatically
  - **FOR JSON PATH** – output will be formatted by the query itself (using aliases)
- Additional options
  - **INCLUDE_NULL_VALUES**
  - **ROOT**
  - **WITHOUT_ARRAY_WRAPPER**

SQL 2 JSON – data conversion

Source data type	Destination data type
Char, Varchar, Nchar, NVarchar, Text, Ntext, Date, DateTime, DateTime2, DateTimeOffset, Time, UniqueIdentifier, Smallmoney, Money, XML, HierarchyId, Sql_Variant	String
Tinyint, Smallint, Int, Bigint, Decimal, Float, Numeric	Number
Bit	Boolean
Binary, Varbinary, Image, Rowversion, Timestamp	Base 64 encoded string
null	null
geography, geometry, and CLR-based user defined data types	**not supported**



SQL 2 JSON

JSON 2 SQL

- **OPENJSON**
- **rowset function** (table-valued function)
- Two types of return tables:

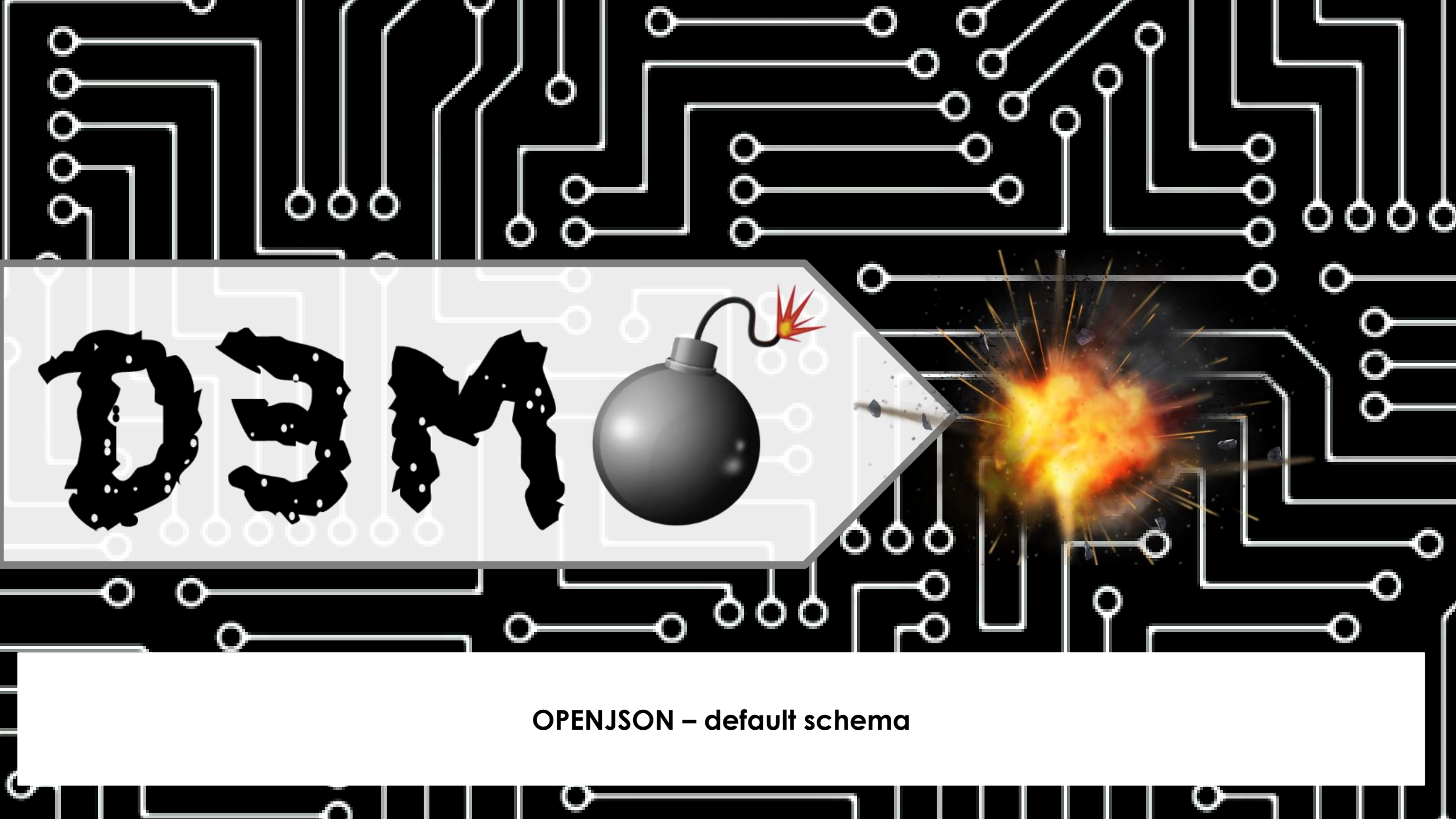
  - **- Default schema** – when the schema is not specified
  - **- Explicit schema** - User-defined schema meaning the returned columns are defined by the user

OPENJSON - default schema

- **OPENJSON (Expression, [Path])**
  - **Expression** – JSON object in Unicode text format
  - **Path** – optional argument to specify a fragment (sub-node) of the input expression

- Return - table result with three columns
  - **Key** – Non nullable NVARCHAR (4000) column that contains the name of a JSON property or the index of the element in the specified array
  - **Value** – Nullable NVARCHAR (MAX) column that contains the value of the property
  - **Type** – TINYINT column representing the data type of the value:

0 -> null	1 -> string	2 -> int	3 -> true/false	4 -> array	5 -> object



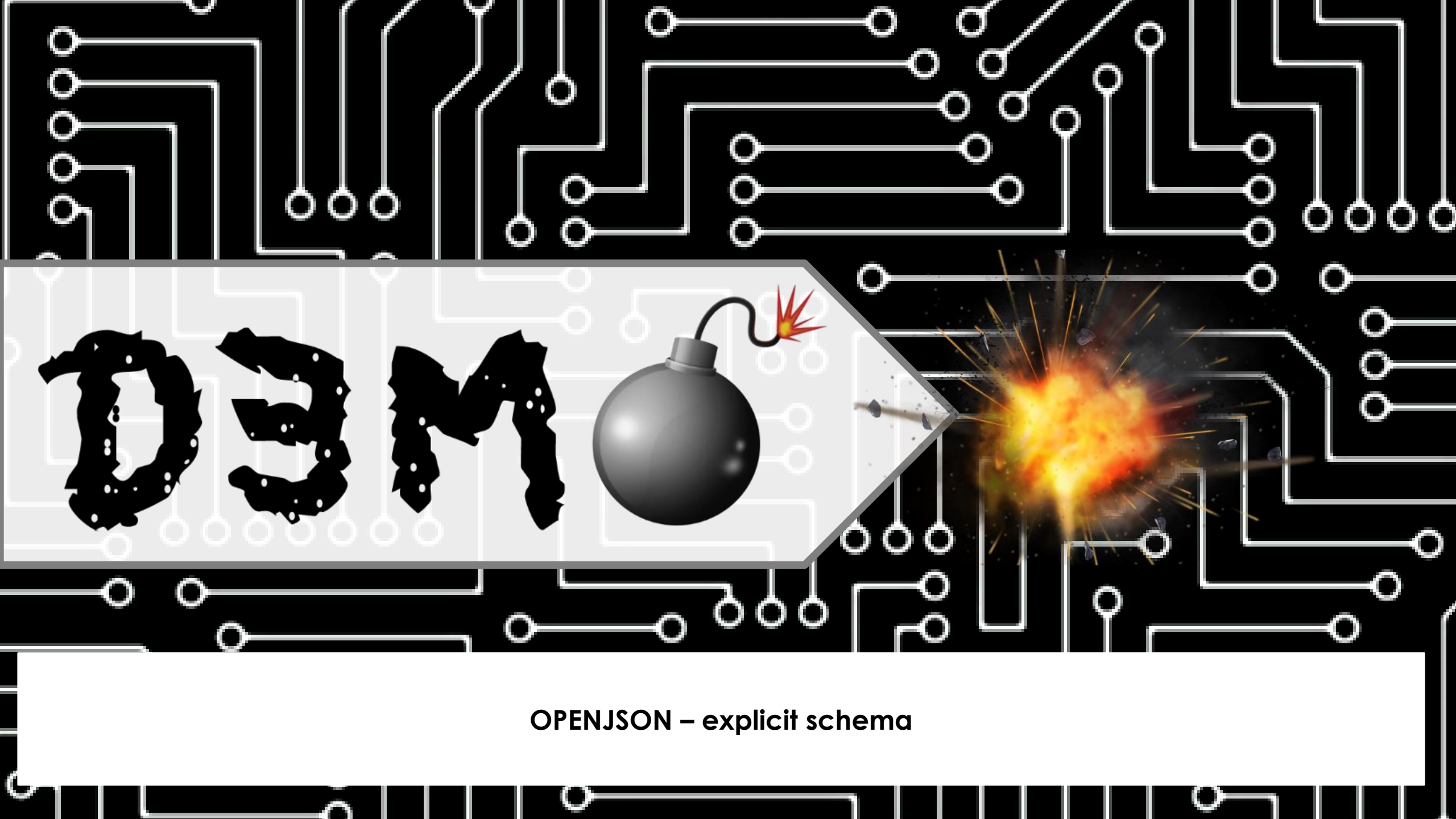
OPENJSON – default schema

OPENJSON - explicit schema

- **OPENJSON (Expression, [Path])**
- [**WITH** (
- **columnName dataType** [**columnPath**] [**AS JSON**]
- [, columnName dataType [columnPath] [AS JSON]]
-)]

- **columnName** – Name of the output column
- **dataType** – Data type of the output column
- **columnPath** – Optional argument to specify a fragment (sub-node) of the column
- **AS JSON** – Optional argument to specify that the referenced property contains an inner JSON object or array. If used, the column must be NVARCHAR(MAX) data type

- WITH keyword - at least one column must be specified!!!



OPENJSON – explicit schema

JSON_VALUE

- extracts a scalar value (primitive data type) from a JSON string
- **JSON_VALUE (Expression, [Path])**
  - **Expression** – JSON object in Unicode text format
  - **Path** – optional argument to specify a fragment (sub-node) of the input expression

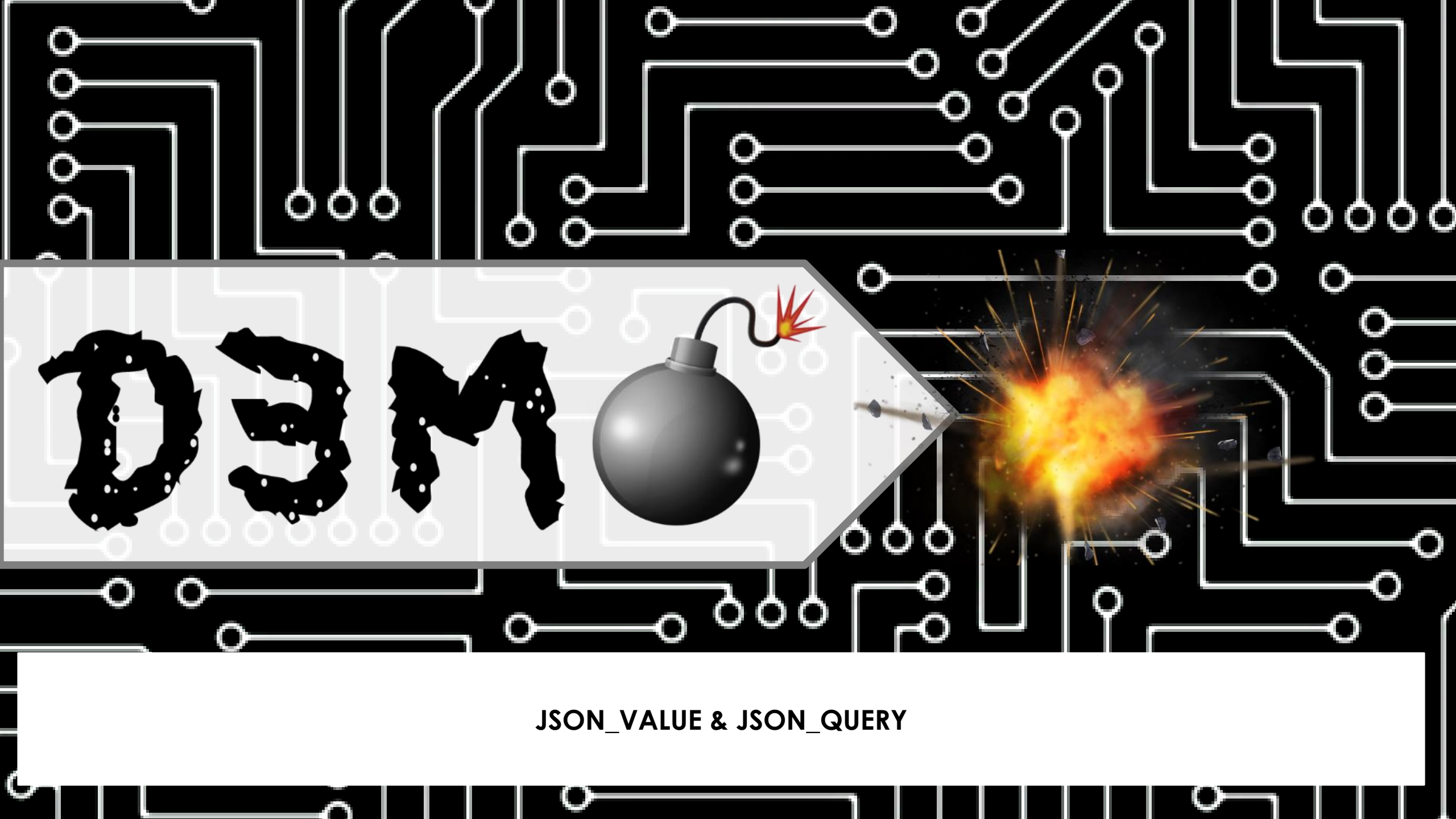
➡ Return – result of nvarchar(4000) data type with the same collation as in the input expression.

- Can be used in SELECT, WHERE, and ORDER BY clauses

JSON_QUERY

- extract a JSON fragment or to get a complex value (object or array)
- **JSON_QUERY (Expression, [Path])**
  - **Expression** – JSON object in Unicode text format
  - **Path** – optional argument to specify a fragment (sub-node) of the input expression

- Return – nvarchar(max) if the input string is defined as (n)varchar(max); otherwise -> nvarchar(4000)



JSON_VALUE & JSON_QUERY

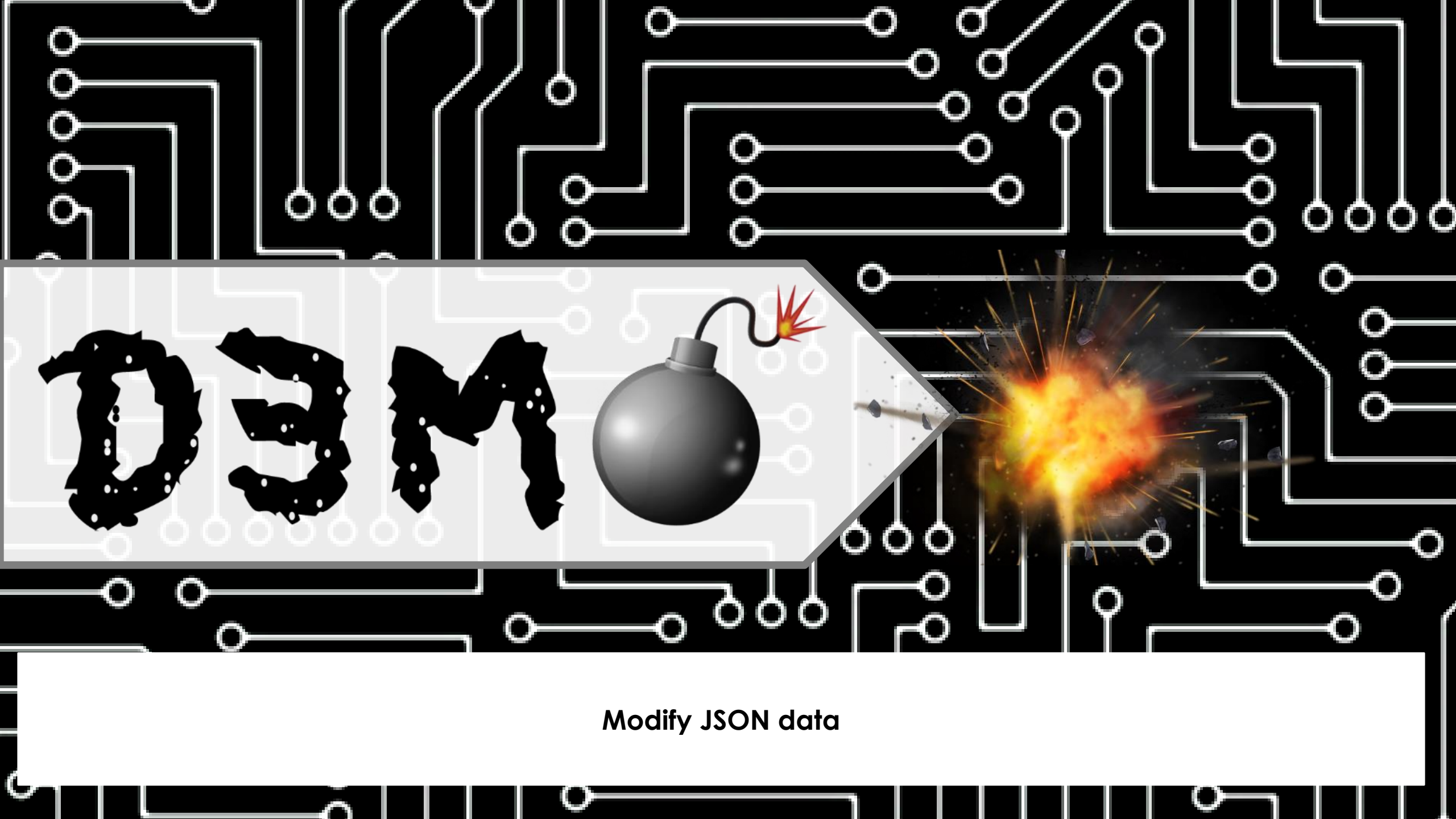
Modifying JSON data

- **JSON_MODIFY (expression , path , newValue)**

  - **Expression** – JSON object in Unicode text format
  - **Path** – A JSON path expression that specifies the property to update
  - **newValue** – The new value for the property specified by path

- Return - updated JSON string

- Adding, Removing, Updating JSON property
- Multiple changes



Modify JSON data

ISJSON

- To JSON or not to JSON ?
- **ISJSON (expression)**

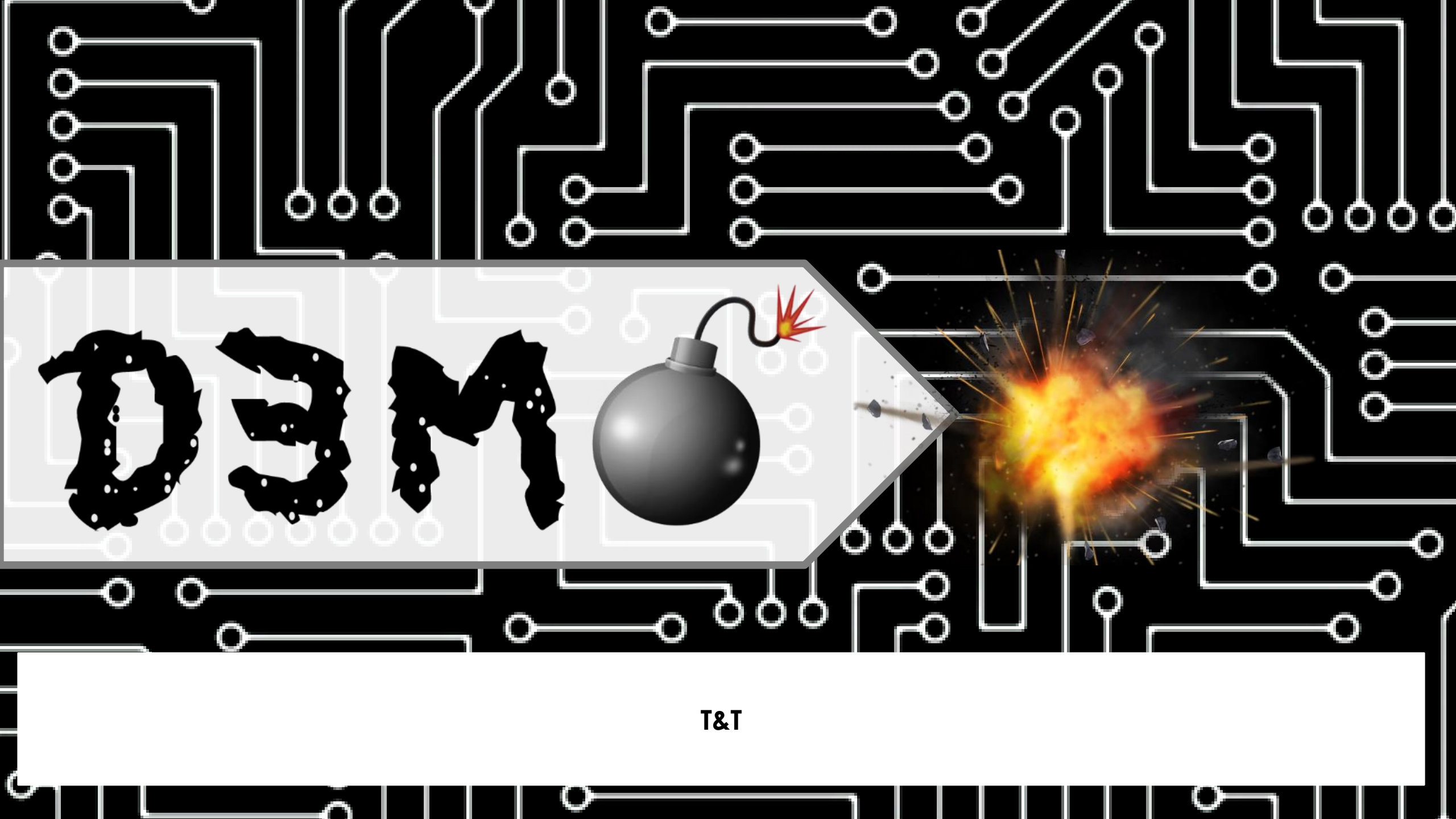
  - **Expression** – The string to test

  - Return – int
    - - 1 - string contains valid JSON
    - - 0 - string is not valid JSON
    - - NULL - input expression is NULL

```
{
    "Name": "John Doe",
    "Name": "John Doe",
    "BlogURL": "http:\/\/www.microsoft.com"
}
```

T&T

- Import JSON from a file
- Indexing JSON data
- Examples:
  - Compare two table rows using JSON
  - Processing data from a comma-separated list of values
  - Hash and compare records
- …



T&T

THE END

Thank you!