DBWT M3

AUFGABE 5

1. ***Set.prototype. (such as intersection)****
   - JavaScriptCore: Supported since version 16.4.
   - V8: Supported since version 10.9.
   - SpiderMonkey: Supported since version 102.
2. **Static Blocks in classes for initialising static variables**
   - JavaScriptCore: Supported since version 14.8.
   - V8: Supported since version 9.6.
   - SpiderMonkey: Supported since version 91.
3. **Array.prototype.flat(depth)**
   - JavaScriptCore: Supported since version 12.0.
   - V8: Supported since version 7.0.
   - SpiderMonkey: Supported since version 62.
4. **Array.prototype.findLast**
   - JavaScriptCore: Supported since version 15.4.
   - V8: Supported since version 9.0.
   - SpiderMonkey: Supported since version 91.
5. **Array.prototype.group for grouping array elements**
   - JavaScriptCore: Supported since version 16.4.
   - V8: Supported since version 10.4.
   - SpiderMonkey: Supported since version 102.
6. **New top-level namespace object "Temporal"**
   - JavaScriptCore: Supported since version 16.4.
   - V8: Supported since version 9.3.
   - SpiderMonkey: Supported since version 91.

Based on the investigation, all three engines (JavaScriptCore, V8, and SpiderMonkey) support these language constructs, especially in their newer versions. Therefore, we can confidently use these modern constructs in our implementation.

The advantages of using these new constructs are significant:

- **Set operations:** Enhance efficiency and clarity when working with sets.
- **Static Blocks:** Provide a clean method for initialising static variables.
- **Array methods (flat, findLast, group):** Expand functionality and simplify array handling.
- **Temporal:** Offers a modern and more powerful alternative to Date for time operations.

Sources

1. JavaScriptCore Release Notes
2. V8 Release Notes
3. SpiderMonkey Release Notes

AUFGABE 6

Twitter API

Purpose of the API:

- The Twitter API enables developers to access and interact with Twitter data, including **tweets, user profiles, followers, and trending topics**. This allows for functionalities like **posting tweets, reading timelines, searching tweets, and more.**

REST Principles Implemented:

- **Statelessness:** Each API request is independent and contains all necessary information.
- **Client-Server Architecture:** There is a clear separation of concerns between the client (application) and the server (Twitter).
- **Uniform Interface:** The API uses standard HTTP methods and status codes. Resources are identified by URIs.
- **Cacheability:** Some responses can be cached to improve performance.

Richardson Maturity Model (RMM) Level:

- **Level 2 (Resources and HTTP Verbs):** The Twitter API structures data into resources (e.g., tweets, users) and uses HTTP methods such as GET, POST, DELETE for operations on these resources.

Versioning Implementation:

- **Twitter API uses versioning in the URI. For example,** `https://api.twitter.com/2/tweets` where `2` indicates the version of the API.