JAVASCRIPT SYNTAX NOTES

TABLE OF CONTENTS:

1. JAVASCRIPT FOUNDATION:

What is JavaScript?

JavaScript is a programming language designed for adding interactivity to websites.

Node.js was later invented to use it in a backend manner also.

2. CORE TECHNOLOGIES:

JavaScript (JS) is one of the 3 core technologies of the web

It is a high-level, integrated programming language used in:
- Client-side web programming
- Server-side web programming

3. GENERAL USES OF JAVASCRIPT:

- DOM manipulation
- Event handlers
- Asynchronous requests
- Animations & effects
- Data visualization & transformation
- Data storage
- Easy-to-use applications (SPA)
- Create APIs or web services (Node.js, Deno)

4. NPM - NODE PACKAGE MANAGER:

NPM allows for locally running JavaScript code.

5. USING THE JAVASCRIPT CONSOLE:

To link JavaScript to HTML:
- In HTML: <p> then ctrl+shift or use mouse to open
- Code can be run directly in the console
- Learn to clear the console & log to the console
- Typing 'console' gives you the list of the console object

6. CONSOLE METHODS:

console.log outputs a value:

```
      console.log(100);
      console.log('Hello World');
      console.log(30, 'Hello', true);
```

Printing with values:

```
      const x = 100;
      console.log(x);
```

console.error:

```
      console.error('Alert');
```

console.warn:

```
      console.warn('Warning');
```

console.table output:

```
      console.table({'name': 'Brad', 'email': 'brad@gmail.com'});
```

console.group:

```
      console.group('simple');
      console.log(x);
      console.error('Alert');
      console.warn('Warning');
      console.groupEnd();
```

7. COMMENTING JAVASCRIPT CODE:

Types of Comments:
----
1. Single-line comment:

```
// A single-line comment
```

2. Multi-line comment:

```
/* A multi-line
            comment */
```

Comments can be used to:

- Explain code one way or off for testing to be done
- Create to-dos (VSP)
- For human use - they have no impact on code that runs

8. CODE BLOCKS:

Use curly braces { } and be consistent in your code style.

Example:

```
if (x > 18) {
      console.log('Adult');
}
```

9. VALUES IN JAVASCRIPT:

- Every piece of information used in a program is a **value**.

- Values transfer into the values implicit in **datatypes**.

Value Types:

- Primitive: number, bigInt, string, boolean, null, undefined, symbol
- Compound: objects, arrays, functions, built-in objects

Working with Values:

- Literals are fixed values you literally provide in code
- Variables are symbolic names that point to values in memory

The typeof operator:

Can be used to determine the type of value

object.values():

The object.values() static method returns an array containing the values of an object's enumerable string-keyed properties.

10. KEYWORDS:

The keywords of JavaScript (in alphabetical order):

await
break
case
catch
class
const
continue
debugger
default
delete
do
else
enum
export
extends
false
finally
for
function
if
import
in
instanceof
let
new
null
return
super
switch
this
throw
true
try
typeof
var
void
while
with
yield

11. IDENTIFIERS:

An identifier is a unique name made from a sequence of characters for entities in code.

Rules of identifiers:

- Must start with a letter, underscore, or dollar sign
- After first character, can contain letters, numbers, underscore or dollar sign
- Cannot be a reserved keyword
- Are case-sensitive

Recommended naming conventions:

- camelCase or SCREAMING_SNAKE_CASE for identifiers(variables or constants).

12. DATA TYPES:

Primitive vs Reference Types:

PRIMITIVE DATA TYPES:
----
1. **string** - sequence of characters, must be in quotes or backticks
2. **number** - integer or floating point
3. **Boolean** - logical true or false
4. **Null** - absence of value
5. **Undefined** - variable not yet assigned
6. **Symbol** - built-in object that returns a unique symbol
7. **BigInt** - numbers too large for the number datatype

Note: Strings, numbers, and Booleans are the most frequently used.

REFERENCE TYPES (objects):

Non-primitive values. When assigning to a variable, the variable is given a reference to that value.

Reference types include:
1. **Object literals**
2. **Arrays**
3. **Functions**
4. **Date**
5. **RegExp**
6. **Map**
7. **Set**
8. **WeakMap** and **WeakSet**

Important Notes:

- JavaScript is dynamically typed/garbage collected. We do not explicitly define the types for variables.

- TypeScript is a superset of JavaScript that is statically typed.

- Primitive types: comparison by value

- Compounds/Reference types: comparison by reference

- datatypes do not have to be explicitly declared, unlike C, C++, Rust, ect.

13. VARIABLES & VARIABLE DECLARATION:

let – const - var: variable keywords

- use let or const in most cases
- var is considered outdated; var has scope issues
- scope: global, local (for later)

Examples:

```
      let firstName = 'Dave';
      let lastName = 'Dave';
      console.log(firstName, lastName);

      let age = 30;
      console.log(age);
```

Variable Naming Rules:

Variables can be **letters, numbers, underscore, and dollar signs**. They cannot start with a number.

Important Note:
----
let can only be used once per variable in a given scope.

However, let can be reused with 2nd variables within a scope. The variable can be recalled without let.

Example:

```
      const PLACE = 'the Netherlands';
      let output = PLACE;
      console.log(output);

      let firstName = 'Fred';
      let lastName = 'Rosev';
      output = firstName + lastName;
      console.log(output);
```

14. LITERALS:

Literals are fixed values written directly into the code. They represent different datatypes that the program inherently understands.

Types of Literals in JavaScript:

NUMERIC LITERALS:
    42, 3.14, -001, 1.23e4

STRING LITERALS:
Enclose in single or double quotes or backticks
    - var1 = “this is a quote”;
    - var2 = ‘this is also a quote’;
    - var3 = `and these are backticks`;

TEMPLATE LITERALS:
    Used for string interpolation using the ${expression} syntax

BOOLEAN LITERALS:
    true/false values

OBJECT LITERALS:
    A list of zero or more pairs of property name/values enclosed in curly braces { }
    - This is a convenient way to make objects
    - Example: const car = { color: 'red', wheels: 4 };

ARRAY LITERALS:
    A list of zero or more values enclosed in square brackets
    - Example: const color = ['red', 'blue', 'green'];

REGULAR EXPRESSION (RegEx):
    Used to define a pattern for matching character combinations in strings.
    They are enclosed with forward slashes.
    - Example: /abc/

NULL LITERAL:
    Represents the intentional absence of an object value.
    - Example: let data = null;

UNDEFINED:
    Not a literal in an operative sense

END OF DOCUMENT