2a Theory JS API pages 13–23: Data Types & Variables

Nicholas Aüf-Bruzzese

July 23, 2025

1 Introduction to Data in JavaScript

Programming in JavaScript revolves around manipulating data, which is essentially any information stored within a program. Data can represent various real-world attributes, such as a person's name, age, hair colour, number of siblings, or location. Understanding how to work with different types of data is fundamental to writing effective JavaScript code. This lesson explores the core data types in JavaScript—numbers, strings, and booleans—and introduces variables as a means to store and manipulate these data types.

2 JavaScript Data Types

JavaScript supports several basic data types, each serving a specific purpose in representing information. The three primary data types discussed here are numbers, strings, and booleans.

2.1 Numbers

Numbers in JavaScript are used to represent numerical values, such as integers or decimals, without distinguishing between types like integers or floating-point numbers. They are essential for calculations and representing quantities like age, height, or any measurable value.

- **Examples:**
  - `5` (an integer)
  - `3.14` (a decimal)
  - `12345` (a larger number)

JavaScript allows you to perform mathematical operations on numbers using operators:

- **Addition (`+`)**: `12345 + 56789` → `69334`
- **Subtraction (`-`)**: `1000 - 17` → `983`
- **Multiplication (`*`)**: `123 * 456` → `56088`
- **Division (`/`)**: `12345 / 250` → `49.38`

Order of Operations: In `1234 + 57 * 3 - 31 / 4`, multiplication and division happen first:
`57 * 3 = 171` and `31 / 4 = 7.75`, so the final result is `1234 + 171 - 7.75 = 1397.25`.

2.2 Strings

Strings represent text in JavaScript. They must be quoted, either single (`'...'`) or double (`"..."`):

- **Examples:**
  - `"Hi, I'm a string"`
  - `"John Doe"`
  - `'example@email.com'`

Common operation:

- `"This is a long string".slice(0,4)` → `"This"`

2.3 Booleans

Booleans are logical values: `true` or `false`. They're used in tests, e.g., `true && false` → `false`.

- **Examples:**
  - `true`
  - `false`

3 JavaScript Statements and Semicolons

Each instruction is a statement, ended by a semicolon (`;`):

- `99 * 123;`
- `"Hello".slice(0,4);`

Including semicolons consistently helps avoid tricky automatic-insertion pitfalls.

4 Variables in JavaScript

Variables are named containers for data.

4.1 Creating Variables

Use `var`:

- `var nick;` Creates `nick` (value: `undefined`).

4.2 Assigning Values

`var age = 12;` sets `age` to 12. Reassign without `var`: `age = 13;` → `13` in the console.

4.3 Using Variables in Calculations

```
var numberOfSiblings = 1 + 3;         // 4
var numberOfCandles   = 8;
var result            = numberOfCandles / numberOfSiblings; // 2
```

4.4 Variable Naming Conventions

- No spaces: `numberOfSiblings` (valid), `number of siblings` (invalid)
- Case sensitive: `myVar` is not equal to `myvar`
- Camel case: `myVariableName`

4.5 Example: Balloon Problem

Incorrect grouping `15 + 9 + 2` → `26`

Correct grouping `(15 + 9) * 2` → `48`

With variables:

```
var guests           = 15 + 9;         // 24
var balloonsPerGuest = 2;
var totalBalloons    = guests * balloonsPerGuest; // 48
```

4.6 More Calculations

Seconds in an hour/day/year and age in seconds:

```
var secondsInMinute  = 60;
var minutesInAnHour  = 60;
var secondsInAnHour  = secondsInMinute * minutesInAnHour; // 3600

var hoursInADay      = 24;
var secondsInADay    = secondsInAnHour * hoursInADay;     // 86400

var daysInAYear      = 365;
var secondsInAYear   = secondsInADay * daysInAYear;       // 31536000
var age              = 23;
var ageInSeconds     = age * secondsInAYear;              // 725328000
```

4.7 Increment/Decrement and Compound Assignment

```
var highFives = 0;
++highFives;  // now 1
highFives++;  // returns 1, then becomes 2
--highFives;  // now 1

var score = 10;
score += 7;   // now 17
score -= 3;   // now 14
```

5 Conclusion

Understanding JavaScript's three primitive types—numbers, strings, booleans—and the use of variables is the foundation for building dynamic programs. Practice in the console to reinforce these concepts.