Type Coercion & Equality

AiCE Software Engineering Foundations — JavaScript & Node.js

Node.js is installed, and you're comfortable declaring variables and working with JavaScript's basic data types. This lesson looks at how JavaScript behaves when different data types mix together — and the one habit that keeps it from catching you by surprise.

1. Type Coercion — JavaScript's Automatic Conversion

In Python, adding a string and a number together stops you with a TypeError. JavaScript does something different — it tries to make the operation work by automatically converting one of the values. This is called type coercion, and it can catch you by surprise if you don't understand it.

Create a file named coercion.js:

```
console.log("5" + 3);
console.log("5" - 3);
console.log("5" * 3);
console.log(true + 1);
console.log(null + 1);
console.log(undefined + 1);
```

Save and run:

Expression	Result	Why
"5" + 3	"53"	The + operator prefers strings — when either side is a string, it joins them together instead of adding.
"5" - 3	2	Unlike +, the - and * operators always try to convert both sides to numbers first.

Expression	Result	Why
"5" * 3	15	Same reasoning as above — both sides are converted to numbers before multiplying.
true + 1	2	JavaScript converts true to the number 1 behind the scenes.
null + 1	1	null is converted to 0 before the addition happens.
undefined + 1	NaN	NaN stands for Not a Number — JavaScript's signal that a mathematical operation was impossible to carry out.

2. Truthy, Falsy, and Strict Equality

Every value in JavaScript is either truthy or falsy when used inside a condition. There are only six falsy values: false, 0, an empty string "", null, undefined, and NaN. Every other value — even "0" as a string, or an empty array — is truthy.

This matters most when comparing values for equality. JavaScript gives two ways to check if two values are equal. Create a file named equality.js:

```
console.log(5 == "5");
console.log(5 === "5");
console.log(null == undefined);
console.log(null === undefined);
```

Save and run: 5 == "5" gives true, but 5 === "5" gives false.

Operator	Called	What It Does
==	Loose equality	Converts the types before comparing — this is why 5 == "5" is true.
===	Strict equality	Compares both the type and the value, with no conversion at all — this is why 5 === "5" is false.

Always Use === and !==

In professional JavaScript, always use === and !==, and never == or !=. Loose equality can give results you don't expect, while strict equality is predictable — no hidden type coercion to reason about.

3. Converting Types Yourself

Instead of leaving conversion to JavaScript's automatic coercion, you can convert types explicitly — this is actually the safer habit. Create a file named datatype_conversion.js:

```
console.log(Number("45000"));
console.log(String(25));
console.log(Boolean(""));
console.log(parseInt("175cm"));
console.log(parseFloat("45.5 per year"));
```

Number() and String() convert a value into a number or a string. parseInt() and parseFloat() are useful when a string has a number mixed with other text — they pull out the number from the front of the string and ignore the rest.

Key Takeaways

JavaScript automatically converts types during operations — type coercion — and the results aren't always intuitive: + prefers joining strings together, while -, *, and other operators convert to numbers first. Truthy and falsy values matter inside conditions, and strict equality (===) is the professional standard over loose equality (==), since it never

silently converts types before comparing. When you need to convert a type on purpose, Number(), String(), Boolean(), parseInt(), and parseFloat() give you explicit control. With coercion and equality understood, the next lesson moves into the two data types you'll use constantly: strings and numbers, along with JavaScript's built-in Math object.