­Jonah Whitney

ADD 147

March 1, 2025

Writing Quality Code

* Debugging
  + Often there are no error messages/ indications where to search for errors.
  + Modern browsers have built in debuggers
  + Console.log() method
    - Displays JavaScript values in the debugger window.
  + Set breakpoints and examine variables while code is executing. JS will stop executing and let you examine values.
  + Debugger keyword – With debugger turned on, the code below stops executing.
* Style Guide
  + Coding conventions are style guides for programming that typically cover naming and declaration rules (vars and functions), rules for use of white space, indentation, and comments, and programming practices/principles.
    - Improves readability and makes code maintenance easier.
  + Variable names
    - Use camelCase for identifier names.
    - All names start with a **letter.**
  + Always put spaces around operators (= + - \* /) and after commas
  + Always use 2 spaces for indentation of code blocks.
  + Always end statements with a semicolon.
  + For complex statements put the opening bracket and end of first line, use one space before opening bracket, put closing bracket on new line, do not end with a semicolon.
  + Object Rules
    - Place opening bracket on same line, use colon plus one space between property/value, no comma after last key-value pair, closing bracket on new line, always end an object declaration with a semicolon.
    - Short objects can be written on one line using spaces between properties.
  + Line length < 80
  + Use simple syntax for loading external scripts
  + Use the same naming convention as JS in HTML
  + Use lowercase file names.
* JavaScript best practices
  + Avoid global variables
  + Always declare local variables w/ var, let, or const keyword otherwise they will become global variables.
  + Declarations on top of each script or function
    - Cleaner code, single place to look for variables, avoid unwanted (implied) global variables, reduce the possibility of unwanted re-declarations.
  + Initialize variables
    - Cleaner code, single place to initialize vars, avoid undefined values.
  + Declare objects with **const –** will prevent any accidental change of type.
  + Declare arrays with **const**
  + JavaScript is loosely typed.
    - A variable can contain all data types.
    - A variable can change its data type.
    - Numbers can accidently e converted to strings or NaN. When doing mathematical operations, numbers can convert to strings.
  + Use === comparison
    - The == comparison operator always converts to matching types before comparing.
    - The === operator forces comparison of values and type. (strict comparison)
  + Avoid Number, String, and Boolean as objects.
    - Treat them as primitive values, not objects. Declaring these types as objects slows down execution speed and produces nasty side effects.
  + End switch block with **default**.
* Common Mistakes
  + Accidently using the assignment operator
    - Using 1 = in an if statement can yield unexpected results.
      * Let x = 0
      * If (x = 10) – returns true because 10 is true, even though x = 0.
      * If(x = 0) – returns false because 0 is false, even though x = 0.
  + Expecting loose comparison
    - == data types don’t matter.
    - === is a strict comparison and type does matter.
  + Switch statements use strict comparison.
  + Confusing addition and concatenation
    - Addition is about adding numbers while concatenation is about adding strings, though both use the “+” operator.
  + Misunderstanding floats
    - All numbers in JS are stored as 64-bits floating numbers. All programing languages have difficulties with precise floating-point values.
    - You can solve problems with floats by multiplying and dividing.
      * Let z = (x \* 10 + y \* 10) / 10
  + JS will allow you to break a statement into two lines, but breaking a statement in the middle of a string will not work. You must use a \ if you must break a statement in a string.
  + Misplacing a semicolon will change the execution of code.
  + If you break a return statement into two lines, it will return **undefined**.
  + Accessing arrays with named indexes
    - If you use a named index when accessing array, JS will redefine the array to a standard object.
    - After the redefinition, array methods and properties will produce undefined or incorrect results.
  + Trailing commas in object and array definition are legal in ECMAScript 5.
  + Undefined is not null
    - Objects, variables, properties, and methods can be undefined.
    - You can test if an object exists by testing if the type is **undefined**.
      * If (typeof obj === “undefined”)
    - You cannot test if an object is **null**, because this will throw an error if the object is **undefined**.
      * If (obj === **null**)
  + JavaScript Performance
    - Reducing activity in loops.
      * BAD – for (let I = 0; I < arr.length; i++)
      * GOOD – let l = arr.length; for (let I = 0; I < l; i++)
      * Bad code accesses the length property in each iteration, while the better code accesses the length property outside the loop and makes the loop run faster.
    - Reduce DOM access
      * If you expect to access a dom element several times, access it once, and use it as a local variable.
    - Reduce DOM size
      * Improves page loading and speed up rendering.
      * Every attempt to search the DOM will benefit from a smaller DOM.
    - Avoid unnecessary variables if you don’t plan to save the values.
    - Delay JavaScript loading
      * Putting scripts at the bottom of the page body lets the browser load the page first.
      * While the sript is downloading, the browser will not start other downloads.
      * An alternative is to use **defer=”true”** in the script tag. Specifies that the script should be executed after the page has finished parsing.
        + Only works for external scripts.
    - Avoid using **with**
  + JavaScript reserved words
    - You cannot use reserved words as variables, labels, or function names.
    - You should also avoid using the name of JavaScript built-in objects, properties, and methods.
    - You should also avoid using the name of HTML and Window objects and properties.
    - JS is often used together with Java, so you should avoid using some Java objects and properties as JS identifiers.
  + Writing Quality JavaScript Code-ii
    - Map, Reduce, and Filter
      * Map – if you want an array of a specific property of an array of objects, map is a better option that **for()** or **forEach()**.
      * Reduce – **reduce()** function returns the aggregate of an array.
      * Filter – **filter()** allows you to filter out some of the elements based on a given parameter.
    - Arrow Syntax
      * Arrow functions are a way of writing concise funtions.
        + => to identify function.
        + Enables the dev to write the same logic in almost half the lines of code.
    - Let, var, and const
      * Var has a functional scope, valid inside the function it is declared in.
      * Let and const cannot be changed.