WW3schools – JavaScript

**JS can:**

* Change HTML content
* Change HTML attributes
* Change HTML styles (CSS)
* Hide elements or Show hidden elements changing the display style

**Display Options:**

* innerHTML: into an HTML element
* document.getElementById(‘id’).innerHTML = ‘Hello’ ;
* document.write(): into the HTML output
* window.alert(): into a dialog box
* console.log(): into the browser console (for debugging purposes)

**SYNTAX:**

* JS values:
  + variables: var is used to declare, = is used to assign
  + literals: fixed values

-Numbers are written with or without decimals

-Strings are text written between double or single quotes

Expressions: combination of values, variables, & operators which computes to a value

Keywords: used to identify actions to be performed

Comments: // or /\* … \*/

Case Sensitive:

* Upper Camel Casing: (Pascal) FirstName;
* Lower Camel Casing firstName;

Statements: instructions to be executed by the web browser

* document.getElementById(‘id’).innerHTML = “Hello”

( tells browser to write ‘Hello’ inside the element with that the given id. )

* Statements can be written on one line when separated by ;
* Ignores multiple white spaces
* Line breaks are best after an operator

**Data Types:**

5 data types that can contain values:

* string
* number
* boolean
* object 🡪 3 types: object, date, array
* function

2 data types that cannot contain values:

* null 🡪 data type is an object
* undefined
* Putting a number is quotes or concatenating a number w/ a string turns the whole statement into a string
* Different sequences produce different results
* X = “Hello” + 2 + 2; 🡪 Hello22
* X = 2 + 2 + “Hello”; 🡪 4Hello
* JS types are dynamic: same variables can be used to hold different data types

**STRINGS:**

* Length of string found using .length property
* Escape characters
* Break up code line in a text string with a \ or use + to concatenate (better)
* Objects cannot be compared

**NUMBERS:**

* Only one type of number; ALWAYS 64-BIT FLOATING POINT
* can be written with or without decimals
* can be written with scientific notation🡪 123e3 = 123000; 123e-5 = 0.00123

Number Methods:

* toString(): returns number as a string;
* Gobal methods for numbers: Number(), parseFloat(), parseInt()

Number Properties: belong to JS’s number object wrapper called Number

**TYPE CONVERSION:** \*ww3schools JS type conversion table\*

Done by the use of a JS function OR automatically by JS itself

* Numbers 🡪 Strings:
* Boolean 🡪 String:
* Dates 🡪 String
* String 🡪 Number:
* Unary + Operator:
* Booleans 🡪 Numbers:
* Dates 🡪 Numbers:
* Automatic Type & String conversion

typeOf Operator: returns the data type of JS variable/expression

* An operator that always returns a string, containing the type of the operand
* Can return 1 of 2 Complex Data Types: function & object

constructor property: returns the constructor function for all JS variables

* can use to check if an object is an Array or Date

**FUNCTIONS:** block of code designed to perform a particular task

* Function name (parameter, parameter) { }
* Function parameters: names listed in the function definition
* Function arguments: real values received by function when invoked
* Function invocation:
* event(user clicks), invoked from JS code, or automatically(self invoked)
* Function Return:
* Return statement, Return value

\*Can be used the same way as variables

* Use a function directly, as a variable value, instead of using a variable to store return value of function

**OBJECTS:** containers for named values called properties or methods

* Object Properties: {name : value pairs} 🡪 {property : property value}
* Access🡪 objectName.propertyName OR objectName[“propertyName”]
* Object Methods: stored in properties as function definitions
* Access🡪 objectName.methodName()

\*Avoid String, Number, & Boolean objects

**SCOPE:**

* Global
* Automatically Global: assigning a value to a variable that hasn’t been declared

( In strict mode, automatically global variables will fail )

* Local: ex. Function arguments (parameters) are local inside functions

**EVENTS:** JS “reacts” (lets you execute code) when HTML events are detected

HTML allows event handler attributes w/ JS code to be added to HTML elements

* <element event= “JS code”>
* <button onclick= “document.getElementById(‘demo’).innerHTML = Date()

( changes content of the element with the id ‘demo’ )

* <button onclick= “this.innerHTML = Date()

( changes the content of its own element )

**BEST PRACTICES:**

* Avoid global variables

Does this mean avoid global everything?

* Always declare local variables; don’t use variables undeclared
* All declarations at the top
* Initialize variables when you declare them
* Never declare Number, String, or Boolean objects
* Don’t use the ‘new Object()’
* Beware of automatic type conversion
* Use the === Comparison
* Use parameter defaults
* End switches with defaults
* Avoid using eval()🡪 poses a security problem

Obvious stuff: operators, identifiers

Random stuff:

A method is actually a function definition stored as a property value