JavaScript Variables

COEN 161

Variables

- In JavaScript *variables* must have unique names
- These unique names are called *identifiers*
- They can be short names like *x* or *y*
- They can be more descriptive like *age, sum,* or *totalValue*
- Reserved words **cannot** be used as identifiers

Assignment

- The = is the primary form of assignment but short hands for other operations exist

Operator	Example	Same As
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

Data Types

- JavaScript supports *loose* data types such as numbers, strings, and objects
- It is important to know what data type you're dealing with

```
var x = 16 + "Volvo";  // This...

var x = "16" + "Volvo"; // ...is the same as this.

var x = 16 + 4 + "Volvo";   // "20Volvo"

var x = "Volvo" + 16 + 4;   // "Volvo164"
```

- JS always takes a string over a number

Data Types

- JavaScript data types are *dynamic*
- The same variable can hold different types

```
var x;               // Now x is undefined

var x = 5;           // Now x is a Number

var x = "John";      // Now x is a String
```

JavaScript Strings

- Can use single or double quotes
- No such thing as chars, only strings
- You can put single quotes inside double, just make sure they match

```
var answer = "It's alright";            // Single quote inside double quotes

var answer = "He is called 'Johnny'";    // Single quotes inside double quotes

var answer = 'He is called "Johnny"';    // Double quotes inside single quotes
```

JavaScript Numbers

- Can be written as integers, decimals, or even scientific notation

```
var x1 = 34.00;     // Written with decimals

var x2 = 34;        // Written without decimals

var y = 123e5;      // 12300000

var z = 123e-5;     // 0.00123
```

JavaScript Booleans

- Can be `true` or `false`

```
var x = 5;

var y = 5;

var z = 6;

(x == y)       // Returns true

(x == z)       // Returns false
```

JavaScript Comparisons

Operator	Description	Example
==	equal to	if (day == "Monday")
>	greater than	if (salary > 9000)
<	less than	if (age < 18)

JavaScript Arrays

- Written in square brackets
- Items separated by commas
- Zero-based index

```
var cars = ["Saab", "Volvo", "BMW"];
```

JavaScript Objects

- Written in curly braces
- Properties are comma separated
- Properties are all a `name:value` pair

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

The typeof Operator

- Used to find the data type of a variable

```
typeof "John Doe"           // Returns "string"

typeof 314                  // Returns "number"
```

Special Types

- Undefined
  - A variable without a value is considered *undefined*

```
var car;  // Value is undefined, type is undefined

car = undefined;   // Value is undefined, type is undefined
```

- Null
  - In JS, null means “nothing”, it’s something that doesn’t exist

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

person = null;        // Now value is null, but type is still an object
```

Undefined vs Null

- They have the same *value* but different types

```
typeof undefined              // undefined

typeof null                   // object


null === undefined            // false

null == undefined             // true
```

Primitive Data

- Single data type with no special properties
  - string
  - number
  - boolean
  - undefined

```
typeof "John"                 // Returns "string"

typeof 3.14                   // Returns "number"

typeof true                   // Returns "boolean"

typeof false                  // Returns "boolean"

typeof x                      // Returns "undefined" (if x has no value)
```

Complex Data

- JS has two complex types
  - function
  - object

```
typeof {name:'John', age:34} // Returns "object"

typeof [1,2,3,4] // Returns "object" arrays are objects

typeof null                  // Returns "object"

typeof function myFunc(){}   // Returns "function"
```

Resources

https://www.w3schools.com/js/js_variables.asp

https://www.w3schools.com/js/js_datatypes.asp