W02 - Programming Basics

- Commenting importance
- Javascript grammer
- Primitive Datatypes
- Strings - String literals, properties and methods
- Declaration and assignment - constants and variables
- Numbers
- Arithmetic
- Undefined and null
- Booleans
- Logical Operators

COMMENTS

// short comment

/* multi-Line longer
comments. can be used to block out code */

GRAMMAR - C-Syntax

ASI allows for lack of semicolon; However Best practice is to use the semicolon (code minifiers, validators)

RESERVED WORDS

- Cannot be used as variable names

PRIMITIVE DATA TYPES

- String - symbol - number - Boolean - Undefined - Null

All data not listed would be an object
- arrays, functions, object literals

typeof operator will return datatype

W02 - Programming Basics

VARIABLES - used to refer to a value stored in memory

Declaring and assigning

const and let are used to declare variables.

const - when value will not be reassigned.

let - when value might be reassigned later in program

```
const name = 'Mike'; // won't be reassigned to another string
let score = 0; // may change
```

Declare and assign multiples:

```
let x = 3, y = 4, z = 5;
```

```
const name = { value: 'Alexa' }; // an object is not immutable
name.value = 'Mike'; // this will update value
```

var ← used in previous versions of javascript, doesn't have same scope limitations as const and let.

Scope - let and const are block scoped, meaning the value only exists within the block in which they are created.

Global Scope - any value declared outside a block

Local Scope - only available within block

NAMING CONSTANTS AND VARIABLES

- use sensible naming conventions
- can start with $, _, and letters. may contain numbers but cannot start with them.
- case sensitive
- camelCase and Underscore_naming

DIRECT ASSIGNMENT AND BY REFERENCE

- Primitive values are direct assignment
- Non-Primitives are by reference

W02 - Programming Basics

```
Const C = {value : 1}
let d = C; // c.value == 1, d.value == 1
d.value = 2; // c.value == 2, d.value == 2.
```

Both variables c and d are referencing the same object. Changing the object affects both c and d.

STRINGS - collection of characters

string literal → 'hello'

String Object → new String ("hello")

- Can use either single or double quote marks
- Use escape (\) to include quote marks in strings 'It\'s me' → It's me

STRING PROPERTIES AND METHODS

- access properties using dot annotation
  - → only object have methods but Javascript creates wrapper objects for primitive types
  - → alternative notation [ ]
- Properties are immutable
- call methods using dot annotation.

```
const name = 'Alexa';
name.toUpperCase(); // name == 'ALEXA'
name.toLowerCase(); // name == 'alexa'
```

TEMPLATE LITERALS - use backtick to delimiinate string

`Hello!` `She said, "It's Me!"`

- Allows use of both double and single quotes within string as well as interpolation of Javascript

```
const name = 'Siri';
`Hello ${name}!`; // prints → Hello Siri!
const age = 39;
`I will be ${age + 1} next year` // prints → I will be 40 next year
```

W02 - Programming Basics

- Template literals can also contain line feed
  → to place a backtick in a template literal, use escape characte

SYMBOLS - new primitive in ES6
- create using Symbol() function

```
const uniqueID = Symbol(); ← use a description inside parenthes
```

object property keys

```
const uniqueID = Symbol('unique ID');
```

NUMBERS - Integers or floating point. Javascript does not distinguish

```
Number.isInteger(); // checks to determine integer or not
    → Number.isInteger(32); ← returns true
    → Number.isInteger(3.14); ← returns false
```

Octal and Hexadecimals

```
0xAF → 175_10 // hexadecimal
0o47 → 39_10  // Octal
0b1010 → 10_10 // binary literal
```

exponents

```
1e6; // 1 * 10^6 = 1000000
2e3; // 2 * 10^3 = 2000
2.5e-3; // 2.5 * 10^-3 = 0.0025
```

Operators

```
2**3; // 2^3 or 8
```

CHANGING VALUES OF VARIABLES - very similar to C++

Infinity - error value for numbers that exceed range allowed in
Javascript: $>1^{308}$ $<-1^{309}$ $1/0$
Smallest number: 5e-324

NaN — error value for Not a Number

CHECK IF VALUE IS NUMBER → Number.isFinite();

Do Not have to explicitly specify datatype

W02 - Programming Basics ## time to order Pens

CONVERTING STRINGS AND NUMBERS

Number('23'); // converts string '23' to 23, if string cannot be converted, returns NaN

String(3); // Converts number 3 to '3'.

10..toString(2); // change base of number to binary → '1010' string

28101..toString(36); // a million in base 36 → 'LoL' string

Parsing Numbers - parseInt() to convert a string representation of a numerical value back to a number
- will use numbers at beginning of string and ignore rest.
- will remove anything after decimal

UNDEFINED AND NULL - null will behave as zero, undefined will error

Booleans - only nine values are always false
- empty strings "" '' ``
- 0
- -0
- NaN
- false
- null
- undefined

LOGICAL OPERATORS - ! - not && - AND
!! - falsy or truthy || - OR

Lazy Evaluation → only check minimum number of criteria

BITWISE - 32-bit integers (32 digit binary numbers (base two)
NOT - converts to negative - 1
AND - 1010 & 1101 = 1000
OR - 12 | 10 = 1100 & 1010 = 1110 = 14
XOR - 12 ^ 10 = 1100 & 1010 = 0110 = 6

W02 - Programming Basics

COMPARISON

Soft Equality '==' double equal signs mean 'is'

Hard Equality '===' only returns true if same datatype

NaN is not equal to itself in Javascript

- Should always use hard equality, converting types before comparison

inequality != soft

!== hard

Sum:

Difference:

Factor:

Quotient: