Week 1 Notes: JavaScript Basics

Ross Emile Aparece

Class 1
01/28/2025

Class 2
01/30/2025

- JavaScript is backwards compatible, have to live with the mistakes in all previouse versions

- JavaScript interpreter is convient for prototyping

- console.log();

  ```
  console.log("Hello World!");
  //prints "Hello World!"
  console.log(5 > 2);
  //prints true
  ```

  – For arrays JS will return the size of the array and the contents of the array

- console.table();

  – returns an xy-table for the array

- Data Types: Primitives and objects

  – 7 Primitives; everything else is an objects
    * number (64 bit IEEE 754 double precision)
      · Range: 10^{127} to $1/10^{127}$
      · Loses precision: $0.1 + 0.2 = 0.30000000000000004$
      · Better to avoid using number and stick with int
      · Stay within max_safe_int and min_safe_int
    * string
      · Same string as in Java
      · Can use " " or ' '
    * boolean
      · True/False
    * undefined
    * null
      · Similar to undefined but can tell apart
      · Undefined when you declare a variable but not assign a value
      · Null is an explicit declaration that a value does not exist
      · Undefined represent a temporary value
    * Symbol (Not discussed)
      · Used for complete uniqueness, ex. primary keys
    * BigInt (Not discussed)

· Used for arbitrarily large numbers
- Primitives cannot be decomposed but objects can
- Objects:
  * arrays
    · Objects with numerical indeces
  * functions
    · First class variables
    · Treated no different than any other data type
  * date/times

- JavaScript uses syntactic sugar

  - Object Wrapper
    * All primitivies have a mirrored object versions
    * JavaScript will casts the primitive to its equivalent object version

- There are 3 ways to declare functions

  - Function Declaration

    ```
    function sum(a,b){
      return a+b;
    }
    sum(3,5);
    //returns 8
    ```

  - Function Expression

    ```
    let sum = function add_two(a,b){
      return a + b;
    }
    sum(10,10);
    //returns 20
    ```

  - Arrow Functions
    * If a function can be represented in a single line use arrow functions
    * Intoduced in 2015 to shorten function expressions

    ```
    let sum = (a,b) => a + b;
    sum(16, 8);
    //returns 24
    ```

    * If single parameter you can skip the parenthesis

```
let sq = x => x * x;
sum(5, 5);
//returns 25
```

* If zero parameters the () must be brought back

```
let hello = () => 'hello';
hello();
//returns hello
```

* Multiline arrow functions are not recommended to use
  · Brings back curly braces and the return statement, losing a
    lot of the usefulness of an arrow function

```
let sum_of_square = (a,b) => {
  let aa = aa;
  let bb = bb;
  return aa + bb;
};
```

- Constructor Functions

  - Input number i: 6
  - Output object employee{eid:i}

```
function new_emp(i){
  return {eid:i};
}
new_emp(5);
//returns eid:5

let new_emp_arr = (i) => {eid:i};
//returns undefined
```

  - Undefined so we have no idea whats wrongs
  - Returns undefined ebcause it matches it with a multiline arrow function
  - Solution:

```
let new_emp_a = (i) => ({eid:i});
new_emp_a(7);
```