Web Advanced: Javascript APIs

"We will learn JavaScript properly. Then, we will learn useful design patterns. Then we will pick up useful tools for making cool things better."

FALL 2018

SESSION #2

SYNTAX, DATA TYPES, OPERATORS, CONDITIONS, LOOPS

jaink@newschool.edu

https://canvas.newschool.edu/courses/1407281

https://classroom.github.com/classrooms/42809645-parsons-web-advanced-javascript-fall-2018

RECAP

JAVASCRIPT SYNTAX

- ➔ Comments
- ➔ Expressions
- ➔ Statements
- ➔ Blocks

COMMENTS

```
// This is my first code
let scheme_map = 42;


/*
 * Here I am building a new plugin that
uses the following arguments:
 * color_code: array()
 * event_name: JSON object
 * returns JSON object
 */


let scheme_map = 42; //used for storage
```

EXPRESSIONS

```
// This is my first code

42

alert("Hello")

"Good" + " " + "Morning"
```

STATEMENTS

```
// This is my first code
let answer = 42;
alert("Hello" + answer);
let greeting = "Good" + " " + "Morning";
```

BLOCKS

```
// This is my first code
{
    let answer = 42;
    alert("Hello" + answer);
    let greeting = "Good" + " " + "Morning";
}
```

DATA TYPES

- ➔ Number
- ➔ String
- ➔ Symbols
- ➔ Booleans
- ➔ undefined/null
- ➔ Auto-type conversion

- ➔ Arrays
- ➔ Functions
- ➔ Objects

NUMBERS

These are all number expressions:

```
42

3.1415

3e8 // 3 x 10^8

4*(12+6)/3

NaN

Infinity / -Infinity
```

STRINGS

These are all string expressions:

```
"Do \"not\" Panic"

"Do not Panic"

'Don't Panic'

'Don\'t Panic'

"We are the Knights Who Say…\n" +

"\"Ni\"!"

`We are the Knights Who Say… //ES6 only

"Ni"!`
```

UNDEFINED/NULL

```
// is this defined?
typeof(varName);

// what is this value?
var nullVariable = null; // null
typeof nullVariable // "object"
```

BOOLEANS

Every value/expression in JS has a Boolean value: true or false.

```
Boolean(expression)

40 > 39 // true

"A" > "B" // false

"a" > "A" // true (lowercase has a higher value)
```

Most values always are TRUE except a few:.

```
False values: "" '' 0 NaN false null undefined
```

TYPE COERCION

Javascript auto-converts the value from one type to another (such as string to number, object to boolean, and so on) as needed to complete an expression:

```
14 + "" // "14"

42 + "0" // "420"

"42" - 7 // 35

"42" * 7 // 294

null || "34" // 34
```

OPERATORS

- ➔ Arithmetic
- ➔ Comparison
- ➔ Logical
- ➔ Assignment
- ➔ Conditional

ARITHMETIC

```
2 + 23;

23 * 5.6;

2 ** 5; //ES8 onwards

46 % 2; //remainder

"23" * 1;
```

COMPARISON

```
5 == 6          // false
5 != 6          // true
"1" == 1        // true
"1" === 1       // false
1 == true       // true
1 === true      // false
```

LOGICAL

Logical operators convert the values into their boolean version (true/false) and then return a boolean value:

```
!30

!null

!!null

20 && 26

20 && -1

3 && 0

true || false || false

NaN || undefined

45 || undefined
```

ASSIGNMENT

```
let a = "5";

let b = 4;

b += a; // b = b + a

b *= a; b = b * a

b %= a; b  = b % a
```

CONDITIONAL

Also known as ternary:

```
(a > b) ? 100 : 200;
```

```
let n = 5;

n % 2 === 0 ? console.log("n is an even
number") : console.log("n is an odd
number");
```

TYPEOF

Takes a single operand only:

```
Number    -> "number"
String    -> "string"
Boolean   -> "boolean"
Object    -> "object"
Function  -> "function"
Undefined -> "undefined"
Null      -> "object
```

```
eg.
typeof(45);
typeof("What am I?");
```

VARIABLES

- ➔ Assignment
- ➔ Typecasting
- ➔ Patterns

VARIABLES

storing a string

```
let my_string = "This is the end.";
```

```
let my_string = new String("This is the end."); // not used much
```

storing a number

```
let my_num = 12;
```

```
let my_num = new Number(12); // not used much
```

ways to define

```
let my_num = 12; // preferred ES6 approach
```

```
var my_num = 12; //traditional and loose
```

```
const my_num = 12; // cannot change it
```

TYPECASTING/COERCION

Javascript auto-converts data types to finish an operation:

```
let answer = "5" * 1;
typeof answer; // number

let answer = "5" + 1;
typeof answer; // string

let answer = true + false;
typeof answer; // number
```

VARIABLES PATTERNS

Group definitions :

```
let my_string, my_num;
let    my_string,
       my_num = 12;
```

Naming patterns (case-sensitive):

```
let $element, first_name;
const CONSTANT;
let functionName();
let ObjectName();
```

VARIABLES PATTERNS

Hoisting - always declare and assign before using:

```
console.log(my_string); // returns error

console.log(my_string); // returns undefined
var my_string;

console.log(my_string); // returns undefined
let my_string = "Hello";
console.log(my_string); // returns Hello
```

PROPERTIES & METHODS

- ➔ Strings
- ➔ Numbers
- ➔ Boolean

STRING PROPERTIES & METHODS

Property:

```
let my_string = "the answer is 42."
console.log(my_string.length)
```

Methods:

```
my_string.slice(0,3);
my_string.charAt(1);
my_string.indexOf("w");
my_string.toUpperCase();
my_string.concat(" ","To be precise","!");
// OR
my_string + " " + "To be precise!"
etc...
```

NUMBER PROPERTIES & METHODS

Methods:

```
let my_number = 3.1415;

my_number.toExponential();

my_number.toFixed(1);

Number.isInteger(my_number);

Number.parseInt(my_number);

etc...
```

CONDITIONS

- ➔ if...else...
- ➔ ternary
- ➔ switch

IF … THEN … ELSE …

```
let age = 23;

if (age == 18) {
  console.log("Sorry, you shouldn't be here.");
}

if (age < 18) {
  alert("Sorry, you shouldn't be here.");
} else {
  console.log("Please proceed.");
}
```

TERNARY

```
let age = 23;

(age < 18) ? console.log("Sorry, you
shouldn't be here.") : console.log("Please
proceed.");


console.log((age < 18) ? Sorry, you shouldn't
be here. : Please proceed.));
```

SWITCH

```
let num = Math.floor(Math.random() * 10);

switch (num) {
  case (4):
    console.log("You rolled a four");
    break;
  case (5):
    console.log("You rolled a five");
    break;
  case (6):
    console.log("You rolled a six");
    break;
  default:
    console.log("You rolled a number less than four");
    break;
}
```

ARRAYS

- ➔ Defining - literal, constructor
- ➔ Common Operations
- ➔ Statements
- ➔ Blocks

DEFINING AN ARRAY

Arrays are special types of objects.

```
// literal approach
const my_array = [];

// constructor approach - not used much
const my_array = new Array();
```

DEFINING AN ARRAY WITH VALUES

```
// prepopulating
const my_array =  ["blue", "red", "green"];


// adding
my_array[0] = "pink";
my_array[3] = "purple";
my_array[5] = null;
my_array[6] = 4;
```

PROPERTIES & METHODS

```
// property
console.log(my_array.length)


// modifiers
my_array.pop(); // updates array
my_array.push(item); // updates array
my_array.concat(second_array); // new array
my_array.join(joiner); // new string
my_array.slice(2,4); // new array starting at
index 2 and ending at index 3
my_array.splice(2,1,"brown");
my_array.includes("brown"); //ES6
```

ARRAY ITERATORS

```
let my_array =  ["blue", "red", "green"];

my_array.forEach(function (val) {
     console.log("Keep working. It's still only " + val);
});

my_array.map(function (val) {
     return (val).toUpperCase();
});

let new_array = my_array.reduce(function (prev, current) {
  return (prev + " " + current);
});
```

LOOPS

- ➔ while
- ➔ do...while
- ➔ for
- ➔ for...in

WHILE LOOPS

Repeat a block of code until a condition remains true:

```
let max_time = 7;
while (max_time < 10){
     console.log("Keep working. It's still only " + max_time);
     max_time++;
}

let max_time = 10;
while (max_time--){
     console.log("Keep working. It's still only " + max_time);
}
```

DO...WHILE LOOPS

Run a block of code at least once and then until a condition remains true:

```
let max_time = 7;

do {

     console.log("Keep working. It's still only " + max_time);

     max_time++;

} while (max_time < 10)
```

FOR LOOPS

Keeps all loop-related vars in one place:

```
for (var max_time = 7; max_time < 10; max_time++) {
     console.log("Keep working. It's still only "+max_time);
}

var my_array =  ["blue", "red", "green"];
for (var i = 0; i < my_array.length; i++) {
     console.log("The selected color: "+my_array[i]);
}
```

NEW in ES6: Use let in block to keep scope local to the loop. eg.

```
for (let i = 0; i < my_array.length; i++) {
     ...
}
console.log(i); //throws error
```

FOR...OF LOOPS

New in ES6 for looping over arrays:

```
let my_array =  ["blue", "red", "green"];
for (const value of my_array) {
     console.log("The selected color: "+value);
}
```

EXAMPLES

Assignment:

Next Steps

- ➔ More Javascript Syntax:
  - datatype: Objects
  - Functions: a special type of Object