

DATA TYPES AND OPERATORS

ADDING JS TO OUR HTML

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>My First Web Page</title>
```

```
    <meta charset="utf-8">
```

```
    <script src="myscript.js" defer></script>
```

```
  </head>
```

```
  <body>
```

```
  </body>
```

```
</html>
```

EXPRESSIONS

- Expressions are combinations literals values, variables or function calls
- All expressions have a type and a value
- Type:
 - Programming languages have types
 - At the lowest level they are identifiers on how something is stored or manipulated by the programming language
 - At a higher level, it signals to the programmer how a value can be used.
 - We can do arithmetic with numbers, but this would not make sense with strings.



TYPES IN JAVASCRIPT

- Primitive types
 - Boolean
 - Number
 - String
 - Null
 - undefined
- Objects
 - Not a primitive type
 - Can be composed of primitive types

WHAT IS AN OPERATOR?

- A symbol or keyword that combines, modifies, or compares one or more expressions
- The result of an operation is also an expression:

`1;`

`1+1;`

`1+1+1;`

- In this example, the plus symbol is the operator
- Any operator that performs an action on two expressions is called a **binary operator**

NUMBERS IN JAVASCRIPT

- What are valid operators?
 - Arithmetic: + - * / %
 - Comparison: < <= > >= !== ===
- + - * /
 - All work the way you think they do with numbers
- % the remainder operator
 - Remember long division?
 - Quotient and *remainder*

NUMBERS IN JAVASCRIPT

- $10 \% 7$
 - what is the remainder after dividing 10 by 7?
 - 7 goes into 10 once (quotient) with 3 left over (remainder)
 - So $10 \% 7$ is 3
- Try it:
- $15 \% 3$
- $10 \% 13$
- $7 \% 4$

NUMBERS IN JAVASCRIPT

- $10 \% 7$
 - what is the remainder after dividing 10 by 7?
 - 7 goes into 10 once (quotient) with 3 left over (remainder)
 - So $10 \% 7$ is 3
- Try it:
- $15 \% 3 = 0$
- $10 \% 13 = 10$
- $7 \% 4 = 3$

NUMBERS IN JAVASCRIPT

- JavaScript treats all numbers the same, though it's useful to distinguish between floating point numbers and integers in programming as there are equivalent concepts in math
- Floating point numbers are approximately the real numbers and integers are the same as integers (whole numbers)

Number Type In Math	Example
Integers	1
Floating point(real)	1.56

NUMBER LITERALS

Base 10

1234

Floating Point

1.23

Sci. Notation

6.7e-11

Special
Values

Value	Description
NaN (Not A Number)	When the result of a calculation does not result in an real number. (Or when, there is not enough information to computer the result)
Infinity	When the result of the calculation is infinity
-Infinity	When the result of the calculation is negative infinity

STRINGS

- Strings are used to store and manipulate text
- They consist of a series of characters wrapped in quotation marks
- examples
 - "I am in CPSC 1045"
 - "123456" (Hint: this is not a number)
 - "jmckeescott01"

STRINGS

- Valid string operators
 - Concatenation (putting strings together)
 - Uses the + operator
 - `"Hello" + " World"`
 - Will become the string
 - `"Hello World"`
 - `55 + "potato"` is `"55potato"`
- What is stored in x?
- `let x = "20" + "17";`

COMPARING VALUES

- Comparisons
 - > >= < <= != ==
 - Comparisons go by 'alphabetical' order relative to the Unicode encoding
- Example:
- `"hello" < "world"`
- `True`
- Because hello alphabetically comes before world

BOOLEANS

- Boolean Data Type has value of true or false
- Valid operators
 - `! && || === !==`
- The **not** operator
 - `!`
 - `!true == false`
 - `!false == true`
 - `5 == 5` is true
 - `5 != 5` is false

BOOLEANS

- && and || are binary operators that combine two Boolean expressions
 - && Is the **and** operator
 - || is the **or** operator
- == and != are binary operators that compare two Boolean expressions

HOW BOOLEAN OPERATORS WORK

- Truth table
- `&&`

- Examples:

- `5 > 4 && 9 > 8`
- `6 > 7 && 8 > 9`
- `8 > 1 && 2 > 3`

a	b	a && b
T	T	T
T	F	F
F	T	F
F	F	F

HOW BOOLEAN OPERATORS WORK

- Truth table
- Ex. or

- Examples:

- $5 > 4 \ || \ 9 > 8$
- $6 > 7 \ || \ 8 > 9$
- $8 > 1 \ || \ 2 > 3$

a	b	a b
T	T	T
T	F	T
F	T	T
F	F	F

VARIABLES

- Variables are storage containers
- In JavaScript they can hold pretty much anything
- Their type changes depending on what is being stored
- We use the let keyword to define a variable
- Ex. `let x;`
 - defines the variable x, with undefined type
- Ex. `let x = 5;`
 - defines the variable x, and places 5 in that container
 - Until I change the value, when I use the symbol x, it will have a value of 5

ASSIGNMENT OPERATORS

- Assignment operators assign (store) a value to a variable
- We've seen the = operator
- But there are several others
 - += -= *= /= %=

Ex.

```
let x = 5;
```

```
x += 10;
```



This is the same as

```
x = x + 10;
```

EXERCISE

- Determine what will be stored in each of the following variables

```
let num1 = "hello" + 2;
```

```
let num2 = 10 % 12;
```

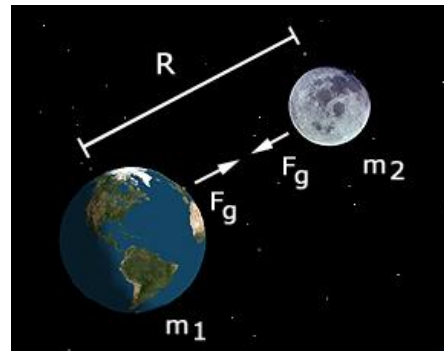
```
let num3 = num2 - 8;
```

EX. COMPLEX NUMERICAL EXPRESSIONS IN JAVASCRIPT

Example:

Newton's Law of Universal Gravitation

```
const G = 6.67e-11;           //declare gravitational constant
let m1 = 5.972e24 //declare mass(kg) of earth
let m2 = 7.347e22 //declare mass(kg) of the moon
let r = 3.84e8;    //declare distance(m) btw earth & moon
let F = G*m1*m2/Math.pow(r, 2);
```



The force of gravity, F_g , is given by

$$F_g = \frac{G m_1 m_2}{R^2}$$

where,

G = gravitational constant = 6.668×10^{-11}

m_1 = mass of object #1

m_2 = mass of object #2

R = distance between the objects

HOW != AND == WORK

- == is the comparison operator
- a == b
 - Returns true if a the same value as b
 - Returns false if a is not the same value as b
 - Ignores type
- != negation of the comparison operator.
- a != b
 - Returns true if a is not the same value as b
 - Returns false if the a is the same value as b

HOW !== AND === WORK

- === is the **exact equal to** comparison operator
- a === b
 - Returns true if a the same as b in both value AND type
 - Returns false if a is not the same as b in either value OR type
- !== negation of the exact comparison operator
- a !== b
 - Returns true if a is not the same as b in either value OR type
 - Returns false if the a is the same as b in both value AND type

EXAMPLES

- `15 === 56;` is false
- `15 <= 56;` is true
- `"Hello" === 456;` is false
- `"456" === 456` is also false
- `"456" == 456` is true
- How about these?
- `"Hi Man" === "Hi" + " Man"`
- `"hi man" == "Hi Man"`

SPECIFICS ABOUT ==

- Different conversions take place depending on the type of the left hand side and right hand side of the statement
- Read the ECMAScript documentation for all the details.
 - The algorithm cannot fit on one slide.
- Briefly:
 - When comparing number to string the Number(<string>) function is applied and then the comparison takes place
 - When comparing number to boolean the Number(<boolean>) function is applied and then the comparison takes place

SPECIFICS ABOUT ===

- In comparisons using === type conversion does not take place
- **Some Interesting cases**
- Result is always false when comparing different types
- When comparing strings, if the strings have the **exact** same characters and length, then the result is true

SO WHICH IS BETTER?

- We have learned that a conversion takes place for ==
- So when you are using **primitive types** stick with ===
- Manually convert to a string or number if you want the == behavior
- This way you will always know what type of variable you are dealing with and you won't have any surprises!

SUMMARY

- Binary Operator joins two expressions
 - **Arithmetic:** + - / * %
 - **Comparison:** < > <= >= === !==
 - Are operators that require two expressions
- Comparison operators compare similar types and return a boolean value
- Assignment operator assigns values/references to variables
 - = += -= *= /= %=

WHAT'S THE DIFFERENCE?

- Boolean, Strings and Numbers are all stored differently inside the computer for efficiency reasons
 - Numbers are stored in a way where it's efficient for computers to do calculations with them
 - Strings are stored in such a way, where it's efficient to manipulate the string
- Boolean, Strings, Numbers are **Primitive** data types in JavaScript
 - That is they are used as part of building blocks in the JavaScript Language
 - JavaScript provides built-in support for these types

CONVERTING STRINGS AND NUMBERS

- This process is called **type conversion**
- Methods to convert strings to numbers
 - `parseInt()`
 - `parseFloat()`
 - `Number()`

```
let x = "13" + "2";  
x = parseInt(x);
```

- What is stored in x?

EXAMPLE

```
let x1 = true;  
let x2 = false;  
let x3 = "999";  
let x4 = "999 888";
```

```
Number(x1) is 1  
Number(x2) is 0  
Number(x3) is 999  
Number(x4) is NaN
```

CONVERTING STRINGS AND NUMBERS

- Methods to convert numbers to strings

Ex. `let x = 5;`

`let str = x.toString();`

- str will now contain "5"

Ex. `let str2 = (5.45).toString();`

- str2 will now contain "5.45"

IDENTIFY VARIABLE TYPE

- The typeof operator returns the type of a value
 - `typeof 1234;` returns "number"
 - `typeof "Hello";` returns "string"
- There are 6 types the typeof can return
 - "string" "number" "boolean"
 - "function" means code
 - "object" means a container class
 - "undefined" means no type (the variable has not been assigned a value)

OPERATOR PRECEDENCE

- AKA Order of Operations
- Terms:
 - Left-associativity (left-to-right)
 - Right-associativity (right-to-left)
- JavaScript Operators are ordered by precedence
 - Operators that have higher precedence will be evaluated first

OPERATOR PRECEDENCE

Common Operator	Precedence	Associativity	Individual operator
Grouping	19	n/a	(...)
Multiplication, Division, Modulo	14	left-to-right	* / %
Addition, Subtraction	13	left-to-right	+ -
Greater than, Less than, etc	11	left-to-right	> < >= <=
Equal, Not equal Exactly equal	10	left-to-right	== != === !==
Logical and	9	left-to-right	&&
Logical or	7	left-to-right	
Assignment	3	right-to-left	=

IMPLICIT TYPE CONVERSION

- Implicit type conversion is when you convert between strings/numbers without specifically using a method to do so
- The results of implicit type conversion will depend on the order of operations
- This is where making mistakes is very easy in JavaScript

Example	Result
"Dog" + 1	Dog1
1 + "Dog"	1Dog
1 + 2 + "Dog"	3Dog
"Dog" + 1 + 2	Dog12
"Dog" + (1 + 2)	Dog3
"Dog" + 2 * 3	Dog6

EXPRESSIONS ARE EVALUATED STEP-BY-STEP

1. The highest precedence operator of evaluated first
 - I. If there is more than one operator with the same precedence, the the associativity rule is used
 - II. That is operators are evaluated either left to right or right to left
2. The operators with the next lower are evaluated next
3. Repeat until you are down to a single value

EXAMPLE

- What will the value of x be after the expression is evaluated?

```
let x = (1+3*4) +3 > "DOG" + "CAT"
```

```
let x =(1+12)+3 > "DOG"+"CAT"
```

```
let x = 13+3 > "DOG" + "CAT"
```

```
let x = 16 > "DOG" + "CAT"
```

```
let x = 16 > "DOGCAT"
```

```
let x = false
```

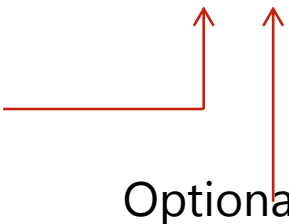
SUBSTRINGS

- We can obtain a portion of a string by using the substring method
- Consider the following string:

```
let testStr = "My name is unknown";
```

- we can extract the word '**name**' by using the substring method.

```
let word = testStr.substring(3,7);
```

Start Index 

Optional end Index

SUBSTRINGS

- If you only specify one index to the substring method
- We can extract the word 'unknown' similarly

```
let testStr = "My name is unknown";
```

```
let word2 = testStr.substring(11);
```

- word2 will now contain "unknown"

EXPRESSIONS IN CONSOLE

- The console window is an interactive JavaScript command prompt
- We can type expressions into the console window and get the results
 - In the browser, right click on the page and go down to "Inspect"
 - In the new area that pops up, find the "Console" tab

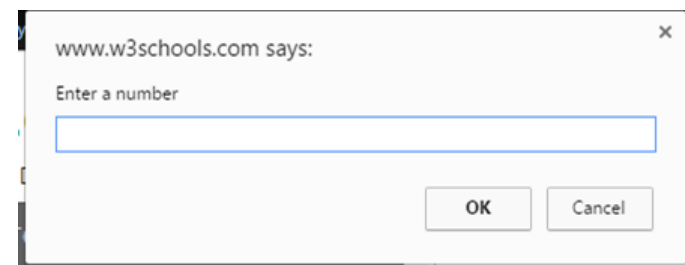
GETTING USER INPUT

- `prompt()`
 - Command used to get user input from a dialog box

- When we write

```
let userInput = prompt('Enter a number') ;
```

- A dialog box will appear, and the JavaScript program will wait until the user enters an input before proceeding
- This is called a **modal** dialog box
- In the above code, what the user types in the dialog box is stored in the variable *userInput as a string* even if the user types a number



DISPLAYING OUTPUT

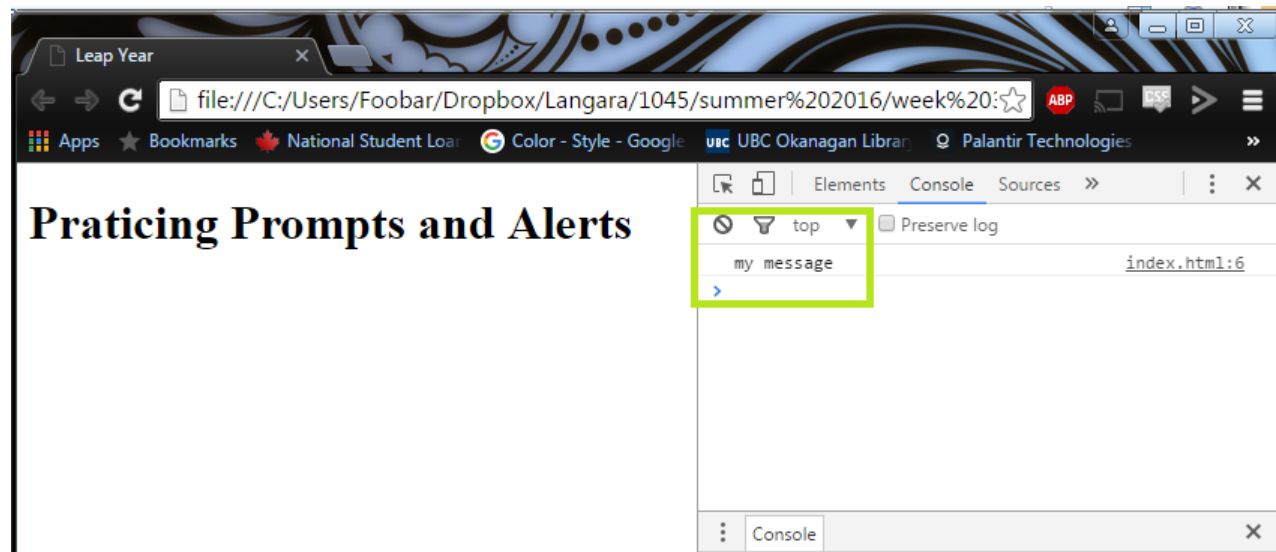
- `alert()` displays a message to the user a message through a dialog box.

```
alert("my message") ;
```



DISPLAYING OUTPUT

- You can also write to the browser console
`console.log("my message");`





HELP CENTER

- Free computer science tutoring in the library
- All day on Tuesdays
- See full schedule at:
<https://langara.ca/student-services/learning-commons/tutoring/computing.html>

EXERCISE

- Write the JavaScript that prompts for the user to enter their first name and last name (separately).
- Then print to the console a string composed of the first letter of the user's first name, followed by the first five characters of the user's last name, followed by a random integer
- You may assume that the last name is at least five letters long.
- Similar algorithms are sometimes used to generate usernames for new computer accounts.
- For example: If the user enters Wayne and Gretzky, your code should print to the console WGretz58 (where 58 could be any random number)