DepED
DEPARTMENT OF EDUCATION

Technical Vocational Livelihood

11

QUARTER

2

Computer Programming

Computer Programming (ICT) – Grade 11
Quarter 2 – Module 14: JavaScript Numbers
First Edition, 2020

Published by the Department of Education Division of Pasig City

Development Team of the Module

Writer : Dan Reinnier C. Amigo
Editor : Ma. Lerma I. Cantanero
Reviewer : Rowena O. Dimagiba
Illustrator:
Layout Artist:
Management Team: Ma. Evalou Concepcion A. Agustin
OIC-Schools Division Superintendent
Dr. Aurelio G. Alfonso
OIC-Assistant Schools Division Superintendent
Dr. Victor Javena
OIC – Chief Curriculum Implementation Division

Education Program Supervisors

Librada L. Agon EdD (EPP/TLE/TVL/TVE)
Liza A. Alvarez (Science/STEM/SSP)
Bernard R. Balitao (AP/HUMSS)
Joselito E. Calios (English/SPFL/GAS)
Norlyn D. Conde EdD (MAPEH/SPA/SPS/HOPE/A&D/Sports)
Wilma Q. Del Rosario (LRMS/ADM)
Ma. Teresita E. Herrera EdD (Filipino/GAS/Piling Larang)
Perlita M. Ignacio PhD (EsP)
Dulce O. Santos PhD (Kindergarten/MTB-MLE)
Teresita P. Tagulao EdD (Mathematics/ABM)

Printed in the Philippines by _____

Department of Education – Division of Pasig City

Office Address: Caruncho Avenue, San Nicolas, Pasig City
Telefax: 641-88-85 / 682-2819
E-mail Address: divisionofpasig@gmail.com

Computer Programming

11

QUARTER 2

MODULE 14

JavaScript Numbers

Writer : Dan Reinnier C. Amigo
Editor : Ma. Lerma I. Cantanero
Validator/Reviewer : Rowena O. Dimagiba

Introductory Message

For the Facilitator:

Welcome to the Computer Programming for the ICT Module on JavaScript Numbers!

This module was collaboratively designed, developed and reviewed by educators from Schools Division Office of Pasig City headed by its Officer-In-Charge Schools Division Superintendent, Ma. Evalou Concepcion A. Agustin in partnership with the Local Government of Pasig through its mayor, Honorable Victor Ma. Regis N. Sotto. The writers utilized the standards set by the K to 12 Curriculum using the Most Essential Learning Competencies (MELC) while overcoming their personal, social, and economic constraints in schooling.

This learning material hopes to engage the learners into guided and independent learning activities at their own pace and time. Further, this also aims to help learners acquire the needed 21st century skills especially the 5 Cs namely: Communication, Collaboration, Creativity, Critical Thinking and Character while taking into consideration their needs and circumstances.

In addition to the material in the main text, you will also see this box in the body of the module:

Notes to the Teacher

This contains helpful tips or strategies that will help you in guiding the learners.

As a facilitator you are expected to orient the learners on how to use this module. You also need to keep track of the learners' progress while allowing them to manage their own learning. Moreover, you are expected to encourage and assist the learners as they do the tasks included in the module.

For the Learner:

Welcome to the Computer Programming for the ICT Module on JavaScript Numbers!

The hand is one of the most symbolized part of the human body. It is often used to depict skill, action and purpose. Through our hands we may learn, create and accomplish. Hence, the hand in this learning resource signifies that you as a learner is capable and empowered to successfully achieve the relevant competencies and skills at your own pace and time. Your academic success lies in your own hands!

This module was designed to provide you with fun and meaningful opportunities for guided and independent learning at your own pace and time. You will be enabled to process the contents of the learning material while being an active learner.

This module has the following parts and corresponding icons:

Expectation - These are what you will be able to know after completing the lessons in the module

Pre-test - This will measure your prior knowledge and the concepts to be mastered throughout the lesson.

Recap - This section will measure what learnings and skills that you understand from the previous lesson.

Lesson- This section will discuss the topic for this module.

Activities - This is a set of activities you will perform.

Wrap Up- This section summarizes the concepts and applications of the lessons.

Valuing-this part will check the integration of values in the learning competency.

Post-test - This will measure how much you have learned from the entire module. Ito po ang parts ng module.

EXPECTATIONS

At the end of the module the learner is expected to:

1. define JavaScript Numbers;
2. differentiate JavaScript strings and numbers;
3. relate Mathematics with JavaScript numbers; and,
4. apply the use of JavaScript numbers on syntax.

PRETEST

Directions: Identify whether the following words are terms relating to mathematical numbers or JavaScript numbers. Write **M** if it is mathematical, **J** if it is JavaScript, and **B** if it relates to both. Write your answer on the space before each number.

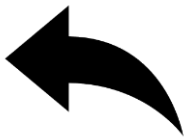
____ 1. cosine

____ 2. tangent

____ 3. Nan

____ 4. -infinity

____ 5. fraction



RECAP

JavaScript Events and Strings were discussed in the previous module. Give a quick description on the following syntaxes:

```
<button onclick="this.innerHTML=Date()">The time is?</button>
```

```
<script>
var x = " J o s e  R i z a l ";
var y = x.length;
document.getElementById("demo").innerHTML = y;
</script>
```

LESSON

JavaScript Numbers

In mathematics, a number is an object used to count, measure, and label. The original examples are the natural numbers 1, 2, 3, 4, and so forth. Numbers are also often used in programming.

In JavaScript, numbers are values that can be used in mathematical operations. You don't need any special syntax for numbers — just write them straight into JavaScript (Example: 12345;).

Decimals, Fractions, Exponents and Negative Numbers

Decimals: Numbers can be written with or without decimals. JavaScript doesn't distinguish between whole numbers and decimals, so you can use them together without having to convert from one to the other.

Syntax:

```
<script>
var x = 3.1415926;  //a number with decimal
var y = 7;          //a number without decimal
var z = x+y;    //addition of numbers with and without decimal
document.getElementById("demo").innerHTML =z;
</script>
```

Output:

10.1415926

Fractions: Fractions do not exist in JavaScript, but you can rewrite them as division problems using the division operator. Note that the resulting number is always converted to decimals. Just like with a calculator, when you write the syntax in fraction the output will be in decimal. Improper fractions use the division operator in the same way.

Syntax:

```
<script>
var x = 1/3;    //a fraction
document.getElementById("demo").innerHTML =x;
</script>
```

Output:

0.3333333333333333

Exponents: Extra large or extra small numbers can be written with scientific (exponent) notation.

Syntax:

```
<script>
var x = 123e5;    // 12300000
var y = 123e-5;   // 0.00123
document.getElementById("demo").innerHTML =x + "<br>" + y;
</script>
```

Output:

12300000
0.00123

Negative Numbers: Negative numbers are created by using a – operator before the number or by subtracting a number from a smaller number.

Syntax:

```
<script>
var x = -3;     // negative 3
var y = 2-5;    // 2 - 5 = -3
document.getElementById("demo").innerHTML =x + "<br>" + y;
</script>
```

Output:

-3
-3

Precision

Integers (numbers without a period or exponent notation) are accurate up to 15 digits.

```
var x = 999999999999999;   // x will be 999999999999999
var y = 9999999999999999;  // y will be 10000000000000000
```

The maximum number of decimals is 17, but floating point arithmetic is not always 100% accurate.

```
var x = 0.2 + 0.1;          // x will be 0.30000000000000004
```

To solve the problem above, it helps to multiply and divide.

```
var x = (0.2 * 10 + 0.1 * 10) / 10;       // x will be 0.3
```

Adding Numbers and Strings

concatenated

JavaScript uses the + operator for both addition and concatenation. Numbers are added. Strings are .

- If you add two numbers, the result will be a number.

```
var x = 10;
var y = 20;
var z = x + y;           // z will be 30 (a number)
```

- If you add two strings, the result will be a string concatenation.

```
var x = "10";
var y = "20";
var z = x + y;           // z will be 1020 (a string)
```

- If you add a number and a string, the result will be a string concatenation.

```
var x = 10;
var y = "20";
var z = x + y;           // z will be 1020 (a string)
```

❖ If you add a string and a number, the result will be a string concatenation.

```
var x = "10";
var y = 20;
var z = x + y;            // z will be 1020 (a string)
```

Other examples:

```
var x = 10;
var y = 20;
var z = "The result is: " + x + y;  // z will be The result is: 1020
```

```
var x = 10;
var y = 20;
var z = "30";
var result = x + y + z;  // z will be 3030
```

The JavaScript interpreter works from left to right. For the first example: the rule of adding a string and a number applies. "*The result is:* " is a string, *x* + *y* is a number.

For the second example, first 10 + 20 is added because x and y are both numbers. Then 30 + "30" is concatenated because z is a string.

Numeric Strings

JavaScript strings can have numeric content.

```
var x = 100;     //x is a number
var y = "10";    //y is a string
```

JavaScript will try to convert strings to numbers in **numeric operations.**

```
var a = "100";
var b = "10";

var c = a / b;       //c will be 10
var d = a * b;       //d will be 1000
var e = a - b;       //e will be 90
var f = a + b;       //f will be 10010 not 110
```

JavaScript uses the + operator to concatenate the strings.

Infinity

Infinity (or **-Infinity**) is the value JavaScript will return if you calculate a number outside the largest possible number.

Division by 0 (zero) also generates infinity.

```
var x =  2 / 0;       // x will be Infinity
var y = -2 / 0;       // y will be -Infinity
```

Syntax:

```
<script>
var myNumber = 2;
var txt = "";
while (myNumber != Infinity) //execute until infinity
{
   myNumber = myNumber * myNumber;
   txt = txt + myNumber + "<br>";
}
document.getElementById("demo").innerHTML = txt;
</script>
```

Output:

4
16
256
65536
4294967296
18446744073709552000
3.402823669209385e+38
1.157920892373162e+77
1.3407807929942597e+154
Infinity

Number Objects

The **Number object** represents numerical date, either integers or floating-point numbers. In general, you do not need to worry about Number objects because the browser automatically converts number literals to instances of the number class.

Numbers can also be defined as objects with the keyword **new.**

```
var y = new Number(123);
```

However, creating Number objects slows down execution speed. In the place of number, if you provide any non-number argument, then the argument cannot be converted into a number, it returns **NaN** (Not-a-Number).

Number Properties

Property	Description
MAX_VALUE	The largest possible value a number in JavaScript can have 1.7976931348623157E+308
MIN_VALUE	The smallest possible value a number in JavaScript can have 5E-324
NaN	Equal to a value that is not a number.
NEGATIVE_INFINITY	A value that is less than MIN_VALUE.
POSITIVE_INFINITY	A value that is greater than MAX_VALUE
prototype	A static property of the Number object. Use the prototype property to assign new properties and methods to the Number object in the current document
constructor	Returns the function that created this object's instance. By default this is the Number object.

Number Methods

The Number object contains only the default methods that are a part of every object's definition.

Method	Description
toExponential()	Forces a number to display in exponential notation, even if the number is in the range in which JavaScript normally uses standard notation.
toFixed()	Formats a number with a specific number of digits to the right of the decimal.
toLocaleString()	Returns a string value version of the current number in a format that may vary according to a browser's local settings.
toPrecision()	Defines how many total digits (including digits to the left and right of the decimal) to display of a number.
toString()	Returns the string representation of the number's value.
valueOf()	Returns the number's value

ACTIVITIES

Creating Syntax: Create your profile. Make sure to utilize the use of JavaScript Numbers and other sub topics discussed in this module. Use the guide below to display the output:

Name	Age	Date	
User name	Birth date	Contact number	
Address			
Course		Section	
Subject	***1st Sem Grade***	***2nd Sem Grade***	***Average***
Mathematics			
English			
Science			
		General Average	

This rubrics serves as basis for scoring.				
	Needs work	**Developing**	**Meets Standard**	**Score**
Content	Minimal learning from the topics were applied	Moderate learning of what was discussed were applied	A lot of learning from the topic were applied	
	5 points	10 points	15 points	

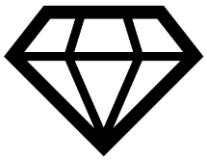
Visual Structure	Visual structure and readability was unorganized	Visual structure was unorganized but somehow readable	Visual structure and readability is well organized	
	5 points	10 points	15 points	
Code	With syntax and logical errors.	Syntax is correct but with logical errors.	There are no logical and syntax errors.	
	10 points	15 points	20 points	
			Total	**/50**

WRAP–UP

This module discussed the JavaScript Numbers. Below are some words or phrases relating to JavaScript Numbers. Discuss each briefly in 2 – 3 sentences.

1. Decimal
2. Fraction
3. Exponent
4. Negative numbers
5. Precision
6. The + operator
7. Numeric String
8. Infinity
9. Number Object
10. Nan

	Very Good	**Good**	**Needs improvement**	**Score**
Knowledge Does your response clearly show you have read and understand the lesson content by correctly defining key terms, key persons and summarizing concepts?	5	3	2	
Analysis Have you clearly stated analysis and give examples to back them up? Does your response provide analysis to the larger concepts of the lesson?	5	3	2	
			Total	**/10**



VALUING

Directions: Read carefully and answer the following questions.

1. Why is it important that we know the difference between Strings and numbers?

   _____
   _____
   _____

2. How can you relate Mathematics with JavaScript Numbers?

   _____
   _____
   _____

3. Why do we need to avoid using number objects? Elaborate.

   _____
   _____
   _____
   _____
   _____

POST TEST

Direction: Using the given variables in each number, give the value for the variable z.

1.	Var x = 5;	var y = 5;	var z = x + y;
2.	Var x = “35”;	var y = 40;	var z = x + y;
3.	Var x = “6”;	var y = “12”;	var z = x – y;
4.	Var x = 10;	var y = “20”;	var z = x * y;
5.	Var x = 1;	var y = 11;	var z = “Programming” + y;

KEY TO CORRECTION

Pre-test:	Post-test:
1. M	1. 10
2. M	2. 3540
3. J	3. -6
4. J	4. 1020
5. B	5. Programming11

References

- "Number" Accessed September 22, 2020 11:00am. https://en.wikipedia.org/wiki/Number
- "JavaScript Numbers" Accessed September 22, 2020 1:45pm. https://www.w3schools.com/js/js_numbers.asp
- "Numbers" Accessed September 22, 2020 2:16pm. https://www.javascript.com/learn/numbers
- "JavaScript - The Number Object" Accessed September 22, 2020 5:00pm. https://www.tutorialspoint.com/java