Client-Side Web GIS

JavaScript

Image from NASA: https://commons.wikimedia.org/wiki/File:Earth_Western_Hemisphere_transparent_background.png#filelinks

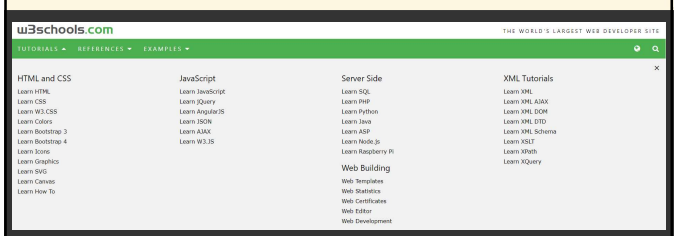
AmericaView
Empowering Earth Observation Education
AmericaView.org - Est. 2003

1

Preparation and Resources

2

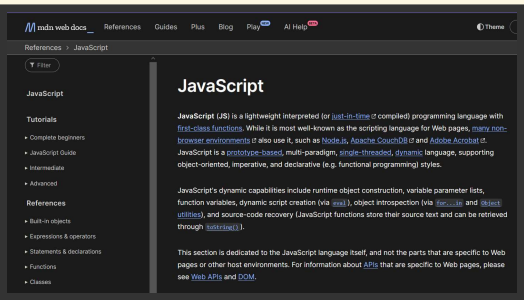
W3schools



https://www.w3schools.com/default.asp

3

Documentation



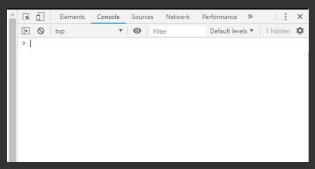
https://developer.mozilla.org/en-US/docs/Web/JavaScript

4

4

Developer Tools

- Ctrl + Shift + I (Chrome or Firefox)
- Can execute JavaScript from the Console
- Can use for debugging



5

5

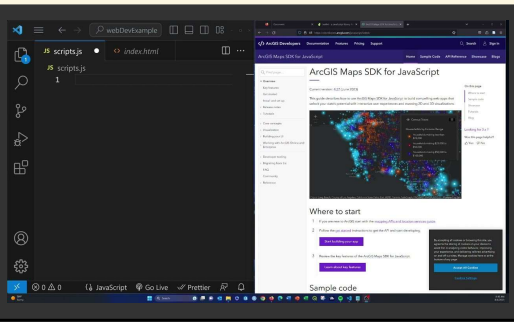
RunJS

https://runjs.app/

6

6

Video: Intro to JavaScript



7

7

Variables, Arithmetic, and Logic

8

8

Declaring and Assigning Variables

- Start with var, let, or const
- = is used for assignment
- End line with ;
- Can hold numbers or strings
- "" or '' around strings
- Can declare and assign in same statement
- Can declare then assign in a second line
- You can declare multiple variables in one statement

- Variable names:
  - must begin with a letter, $, or _
- Can contain letters, digits, underscores, and dollar signs
- Names are case sensitive
- Reserved words cannot be used

```
let x = 2;
let y = "Google Earth Engine";
let z = "2";
```

```
let x;
x = 2;
```

9

9

console.log()

```
console.log(x);
console.log(y);
console.log(z);
```

```
>> let x = 1;
   console.log(x);

   1
```

10

10

Scope

- Local Scopes: variables are declared within a function and can only be called in the function
- Global Scope: variables are declared outside of a function and can be called anywhere in the code

11

11

var, let, const

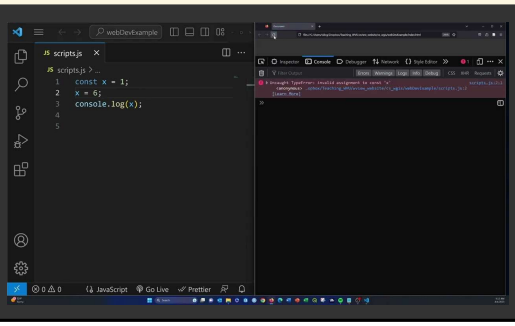
```
var x = 1;
let y = 1;
const z = 1;
```

Declaration Keyword	Scope	Re-Declared/Update	Hoisting
var	Global when declared outside of a function; local when declared inside of a function	Yes/Yes	Hoisted to top of scope and initialized as undefined
let	Global when declared outside of a block; local when declared inside of a block	No/Yes	Hoisted to top of scope but not initialized
const	Global when declared outside of a block; local when declared inside of a block	No/No	Hoisted to top of scope but not initialized

12

12

Video: Variables



13

13

Comments

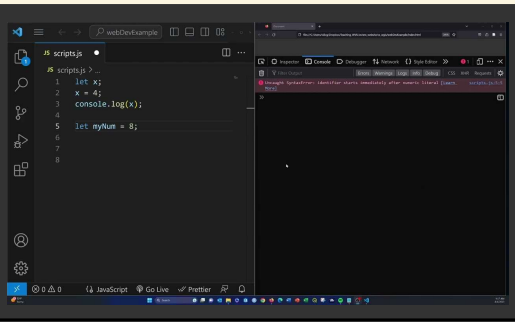
- Comments are meant for humans, and will not be executed by the machine
- Comment your code!!!!

```
//Comment one own line
const x = 4; //Comment after code
/*
Multi-line
comment.*/
```

14

14

Video: Comments



15

15

Strict Mode

https://www.w3schools.com/js/js_strict.asp

16

16

Arithmetic Operators

Operator	Use
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation
/	Division
%	Modulus
++	Increment
--	Decrement

17

17

Arithmetic

```
let x = 11;
let y = 7;
let z = x + y;
console.log(z);

let x = 11;
let y = 7;
let z = x - y;
console.log(z);

let x = 11;
let y = 7;
let z = x * y;
console.log(z);
```

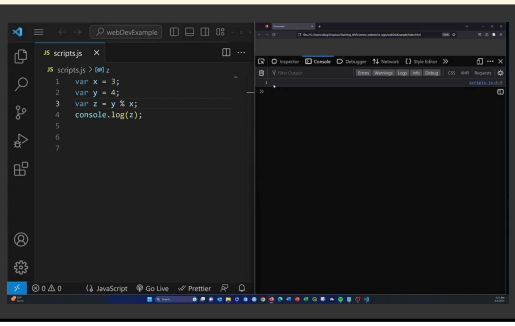
```
let x = 11;
let y = 7;
let z = x / y;
console.log(z);

let x = 11;
let y = 7;
let z = x % y;
console.log(z);

let x = 11;
let y = 7;
let z = x ** y;
console.log(z);
```

```
let x = 5;
x++;
let z = x;
console.log(z);

let x = 5;
x--;
let z = x;
console.log(z);
```

18

18

Video: Operators



19

19

Assignment

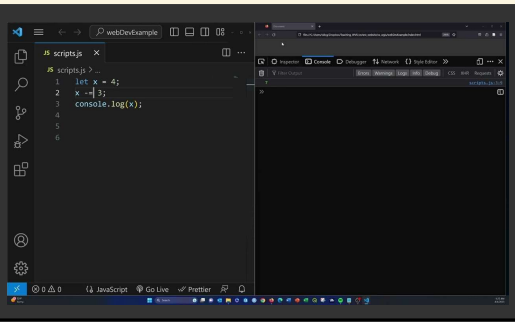
```
let x = 2;
x += 3;
console.log(x);

let x = 2;
x -= 3;
console.log(x);
```

```
let x = 2;
x *= 3;
console.log(x);

let x = 2;
x /= 3;
console.log(x);

let x = 2;
x %= 3;
console.log(x);
```

20

20

Video: Variable Assignment



21

21

Comparison Operators

Operator	Use
==	Equal to
===	Equal value and equal type
!=	Not equal to
!==	Not equal value or not equal type
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

22

22

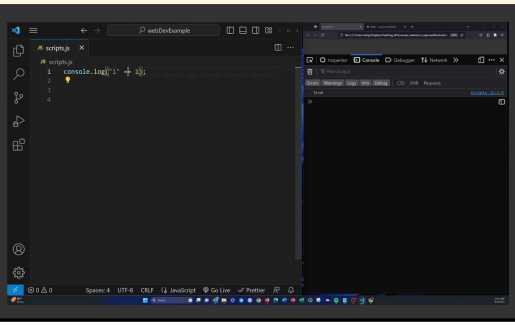
Logical Operators

Operator	Use
&&	And
||	Or
!	Not

23

23

Video: Comparison and Logical Operators



24

24

Data Types

25

25

Data Types

Numbers

```
let x = 2;
let y = 1.2;
```

Strings

```
let x = "Remote Sensing";
let y = "1";
```

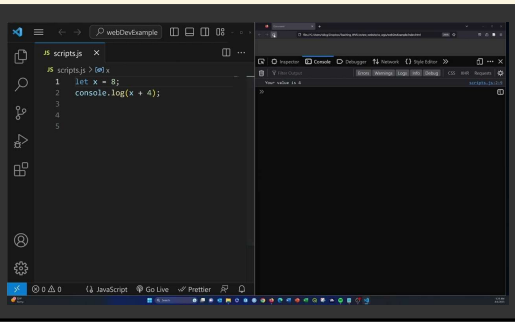
Array

```
let a = ["GIS", "Remote Sensing", "Cartography"];
let a = ["text", 1, true, [1,2,3]];
```

26

26

Video: Numbers and Strings



27

27

Printing Strings

Print string

```
x = "Geog"
y = 455
z = "Remote Sensing"
console.log("The name of the course is " + x + " " + y + ": " +
z + ".");
```

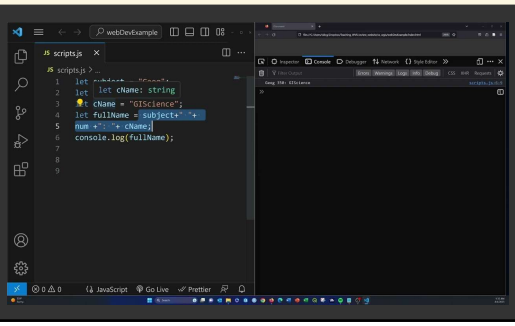
Print string using template literal

```
x = "Geog"
y = 455
z = "Remote Sensing"
console.log(`The name of the class is ${x} ${y}: ${z}.`);
```

28

28

Video: String Concatenation and Printing



29

29

Indexing for Arrays

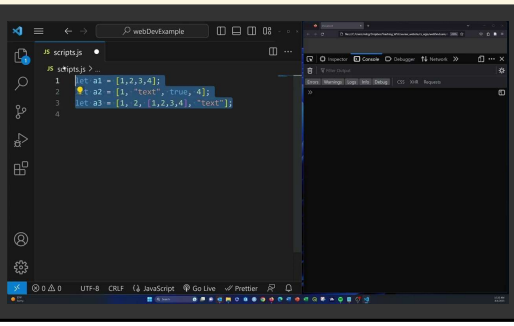
- Indexing starts at 0 as opposed to 1.

```
let a = ["text", 1, true, [1,2,3]];
let a1 = a[0];
let a2 = a[1];
let a3 = a[2];
let a4 = a[3][0];
console.log(a1);
console.log(a2);
console.log(a3);
console.log(a4);
```

30

30

Video: Arrays



31

31

Data Types

Boolean

```
let x = true;
let y = false;
```

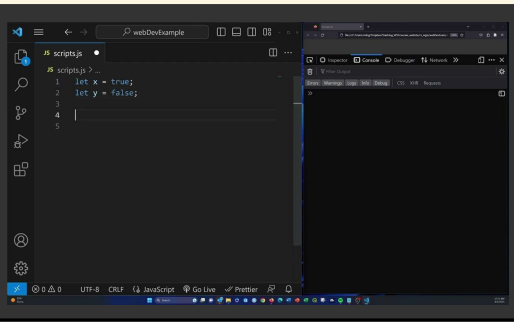
Object

```
let course = {
  code: "Geog",
  number: 456,
  name: "Remote Sensing Applications"
};
```

32

32

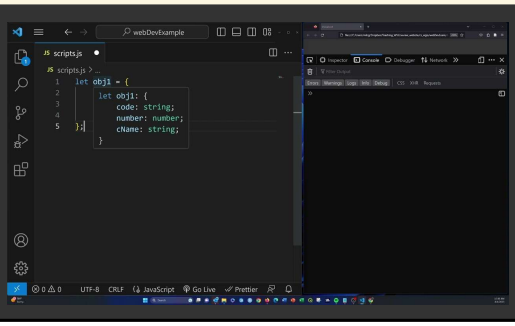
Video: Boolean



33

33

Video: Objects



34

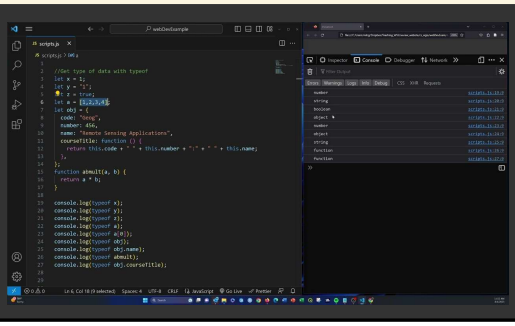
Checking data types

```
let x = 1;
let y = "1";
let z = true;
let a = [1,2,3,4];

let obj = {
  code: "Geog",
  number: 456,
  name: "Remote Sensing Applications",
  courseTitle: function () {
    return this.code + " " + this.number + ":"
+ " " + this.name;
  },
};


function abmult(a, b) {
  return a * b;
}
```

```
console.log(typeof x);
console.log(typeof y);
console.log(typeof z);
console.log(typeof a);
console.log(typeof a[0]);
console.log(typeof obj);
console.log(typeof obj.name);
console.log(typeof abmult);
console.log(typeof obj.courseTitle);
```

35

Video: Check or Get Data Type



36

Data Types

- Types are dynamic for a variable
- Strings can be written in double or single quotes
- Numbers written in quotes will be treated as strings
- If you have quotes as part of the string, use the other type of quote to define the object as a string
- Numbers can be written using scientific notation
- Arrays require square brackets
- Objects are written with curly brackets
- Objects have properties
- Functions are available to convert between data types

37

37

Truthy and Fasly

- Numbers
  - Translate to Boolean true: any non-zero number
  - Translate to Boolean false: 0, NaN, null, undefined
- Strings
  - Translate to Boolean true: any non-empty string
  - Translate to Boolean false: an empty string

38

38

Functions

- Function do something when called
- Functions can be assigned a name and they accept parameters
- In order to execute a function, you will provide arguments for each parameter
- Return statements are used to stop the function and have it return something
- Variables declared within a function are local variables

```
function abmult(a, b) {     let x = abmult(4, 7);
  return a * b;
}
```

39

39

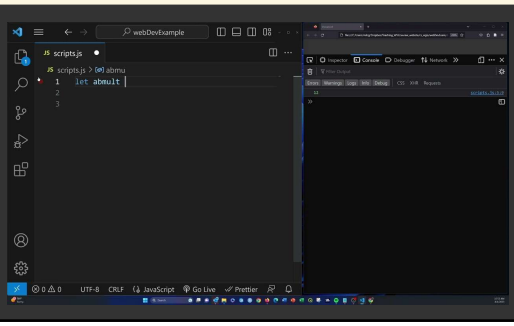
Arrow Functions

```
let abmult = (a, b) => a*b;
let x = abmult(4, 7);
```

40

40

Video: Functions



41

41

Working with Objects

- Objects allow you to associate many values with a single variable
- Values are written as name:value pairs called properties
- Breaks are not important
- Access properties:
  - course.code
  - course["code"]

```
console.log(course.code);
console.log(course["code"]);
```

```
let course = {
  code: "Geog",
  number: 456,
  name: "Remote Sensing Applications"
};
```

42

42

Working with Objects

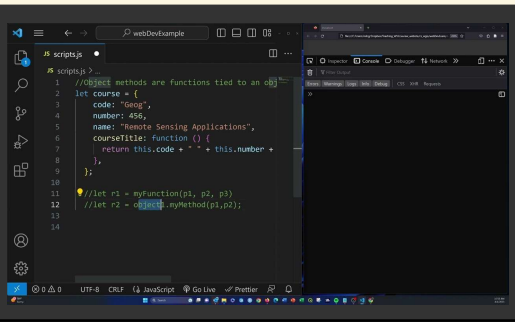
```
let course = {
  code: "Geog",
  number: 456,
  name: "Remote Sensing Applications",
  courseTitle: function () {
    return this.code + " " +
this.number + ":" + " " + this.name;
  },
};


console.log(course.courseTitle());
```

- Methods are actions that can be performed by objects
- Methods are stored in properties as function definitions
- this is a keyword that means the property from this object
- Access a method:
  - course.courseTitle()
- Call without () will return the function definition

43

43

Video: Methods



44

44

JavaScript Classes

- Blueprint for objects of the same type
- Use constructor method to create a new object of this class
- Pre-defined methods, properties, and options to work with for:
  - Creation
  - Interaction
  - Events
  - Get, set, and change settings and options

45

45

Working with Strings

- Backslash (\) escape character turns special characters into string characters

```
let x = 'That\'s mine!';
```

46

46

Working with Strings

Operation	Use
length	Returns string length
indexOf()	Returns index of first occurrence of specified text
lastIndexOf()	Returns the index of the last occurrence of a specified test in a string
search()	Search for specified value and returns the index
slice()	Returns a portion of a string
substring()	Like slice() but cannot accept negative indexes
substr()	Like slice() but can specify length
replace()	Replaces a specified value with another value
toUpperCase()	Converts to upper case
toLowerCase()	Converts to lower case
concat()	Joins two or more strings
charAt()	Returns character as a specified index
split()	Convert string to an array

https://www.w3schools.com/js/js_string_methods.asp

47

47

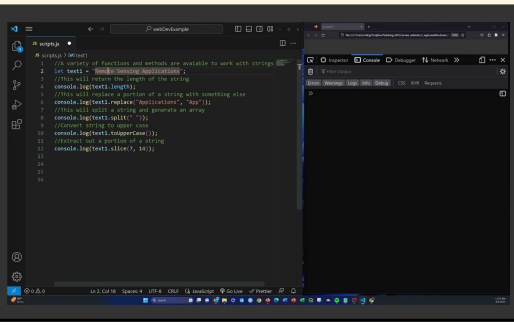
Working with Strings

```
let text1 = "Remote Sensing Applications";
console.log(text1.length);
console.log(text1.replace("Applications", "App"));
console.log(text1.split(" "));
console.log(text1.toUpperCase());
console.log(text1.slice(7, 14));
```

48

48

Video: String Methods



49

49

Working with Numbers

Operation	Use
toString()	Convert number to string
toExponential()	Returns a string using exponential notation with rounding
toFixed()	Returns a string with specified number of decimal places
toPrecision()	Returns a string of a specific length
valueOf()	Returns a number as a number
Number()	Returns a number
parseFloat()	Returns floating point number
parseInt()	Returns integer number

https://www.w3schools.com/js/js_number_methods.asp

50

50

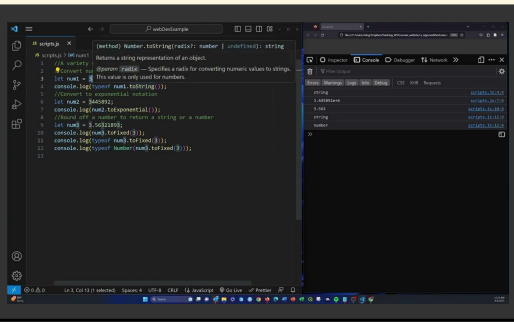
Working with Numbers

```
let num1 = 3;
console.log(typeof num1.toString());
let num2 = 3445892;
console.log(num2.toExponential());
let num3 = 3.56321893;
console.log(num3.toFixed(3));
console.log(typeof num3.toFixed(3));
console.log(typeof Number(num3.toFixed(3)));
```

51

51

Video: Number Methods



52

52

Math Object Methods

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y, x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Returns the value of x rounded up to its nearest integer
cos(x)	Returns the cosine of x (x is in radians)
exp(x)	Returns the value of E^x
floor(x)	Returns the value of x rounded down to its nearest integer

https://www.w3schools.com/js/js_math.asp

53

53

Math Object Methods

Method	Description
log(x)	Returns the natural logarithm (base E) of x
max(x, y, z, ..., n)	Returns the number with the highest value
min(x, y, z, ..., n)	Returns the number with the lowest value
pow(x, y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Returns the value of x rounded to its nearest integer
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle

https://www.w3schools.com/js/js_math.asp

54

54

Math Object

```
console.log(Math.PI);
console.log(Math.E);
```

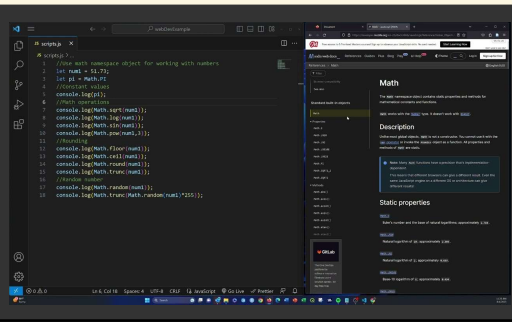
```
console.log(Math.trunc(3.8));
console.log(Math.ceil(3.8));
console.log(Math.floor(3.8));
console.log(Math.round(3.8));
console.log(Math.random(3.8));
console.log(Math.abs(-8.1));
console.log(Math.sqrt(8.1));
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math

55

55

Video: Math Object



56

56

Working with Arrays

Operation	Use
join()	Joins all array elements into a string
pop()	Removes the last element from an array
push()	Adds a new element to end of an array
shift()	Removes first array element
unshift()	Adds new element to an array at the beginning
splice()	Add new items to an array at specified index
concat()	Merges arrays into a new array
slice()	Slices out a piece of an array
toString()	Convert array to string
reverse()	Sorts an array alphabetically
Math.max.apply()	Find the highest number in an array
Math.min.apply()	Find the lowest number in an array

https://www.w3schools.com/js/js_array_methods.asp

57

57

Working with Arrays

```
let arr1 = ["Geography:", "456", "Remote", "Sensing",
"Applications"];

console.log(arr1.join(" "));
arr1.push("Spring", "2019");
console.log(arr1);
arr1.reverse();
console.log(arr1);
let numarr = [1, 3, 5, 8, 9];
console.log(Math.max.apply(null, numarr));
```

58

58

Working with Arrays

Operation	Use
forEach()	Calls a function once for each array element
map()	Creates a new array by performing a function on each array element
filter()	Creates a new array from elements that pass a test
reduce()	Runs a function on each array element to produce a single value
reduceRight()	Like reduce() but works from right-to-left
every()	Check if all array values pass a test
some()	Check if some array values pass a test
indexOf()	Search an array for an element value and returns its position
lastIndexOf()	Like indexOf() but starts are the end of an array
find()	Returns the value of the first array element that passes a test
findIndex()	Like find() but returns the index

https://www.w3schools.com/js/js_array_methods.asp

59

59

Working with Arrays

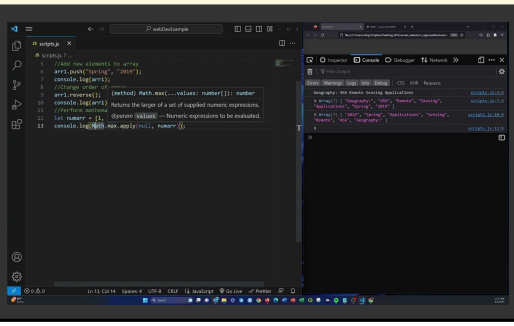
```
let num1 = [3, 4, 5, 6, 7, 8];
function addone(value) {
  return value + 1;
}
let num2 = num1.map(addone);
console.log(num2)

let num1 = [3, 4, 5, 6, 7, 8];
function addgt6(value) {
  return value > 6;
}
```

```
let num2 =
num1.filter(addgt6);
console.log(num2);

let num1 = [3, 4, 5, 6, 7, 8];
function addgt6(value) {
  return value > 6;
}
let num2 = num1.every(addgt6);
console.log(num2)
```

60

60

Video: Array Methods



61

61

Working with Dates

```
let date1 = new Date();
console.log(date1.getFullYear());
console.log(date1.getTime());
console.log(date1.getHours());
console.log(date1.getMonth());

let today = new Date();
let secondDate = new Date();
secondDate.setFullYear(2200, 11, 18);
console.log(today > secondDate);
```

62

62

Working with Dates

Operation	Use
getFullYear()	Get year from date (yyyy)
getMonth()	Get month from date (0-11)
getDate()	Get day as number from date (1-31)
getHours()	Get hours from date (0-23)
getMinutes()	Get minutes from date (0-59)
getSeconds()	Get seconds from date (0-59)
getMilliseconds()	Get milliseconds from date (0-999)
getTime()	Get the time (milliseconds since January 1, 1970)
getDay	(Get the weekday as a number (0-6)
Date.now()	Get the time

https://www.w3schools.com/js/js_date_methods.asp

63

63

Working with Dates

Operation	Use
setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

https://www.w3schools.com/js/js_date_methods_set.asp

64

64

Working with Booleans

```
let x = 3 > 9;
console.log(x);

let x = 2;
let y = "2";
console.log(x == y);
console.log(x === y);
```

65

65

Control Flow, Loops, and Switches

66

66

Control Flow

```
let x = 8;
if (x < 10) {
  console.log("Value is less than 10.");
}
let x = 12;
if (x < 10) {
  console.log("Value is less than 10.");
}
x = 8;
if (x >= 10) {
  console.log("Value is greater than or equal to 10.");
} else {
  console.log("Value is smaller than 10.");
}
```

67

67

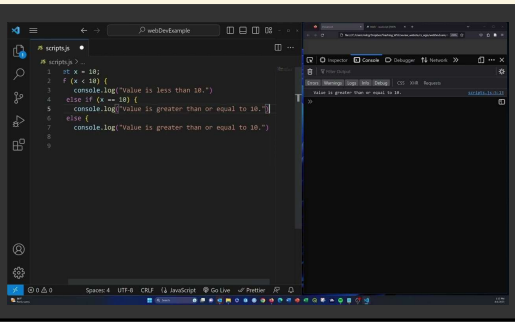
Control Flow

```
let x = 8;
if (x === 10) {
  console.log("Value is 10.");
} else if (x < 10) {
  console.log("Value is less than 10.");
} else {
  console.log("Value is larger than 10.");
}
```

68

68

Video: Control Flow



69

69

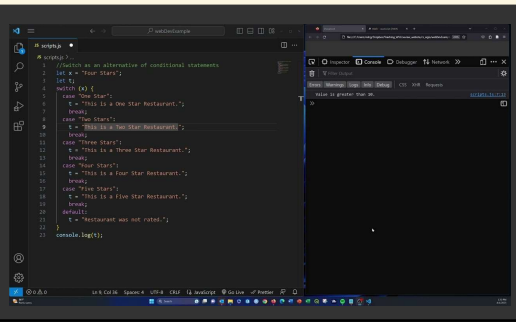
Switch

```
let x = "Four Stars";
let t;
switch (x) {
  case "One Star":
    t = "This is a One Star Restaurant.";
    break;
  case "Two Stars":
    t = "This is a Two Star Restaurant.";
    break;
  case "Three Stars":
    t = "This is a Three Star Restaurant.";
    break;
  case "Four Stars":
    t = "This is a Four Star Restaurant.";
    break;
  case "Five Stars":
    t = "This is a Five Star Restaurant.";
    break;
  default:
    t = "Restaurant was not rated.";
}
console.log(t);
```

70

70

Video: Switch



71

71

For Loop

```
for (i = 0; i < 10; i++) {
  console.log("The value is " + i + ".");
}

for (i = 0; i < 10; i++) {
  console.log(`The value is ${i}`);
}
```

72

72

For Loop

```
let countries = ["Belgium", "Germany", "Iceland",
"Hungary", "Scotland"];

let i = 0;
let len = countries.length;
for (; i < len; i++) {
  console.log("I would like to visit" + " " +
countries[i] + ".");
}
```

73

73

For Loop + Flow Control

```
for (i = 0; i < 15; i++) {
  if (i == 10) {
    console.log("Value is 10.");
  } else if (i < 10) {
    console.log("Value is less than 10.");
  } else {
    console.log("Value is larger than 10.");
  }
}
```

74

74

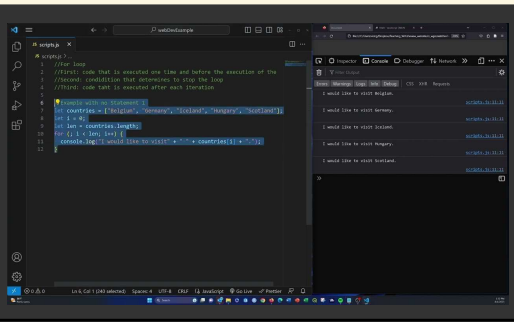
For Loop + Flow Control

```
a = [1,2,3,4,5,6,7,8,9];
let i = 0;
aLen = a.length;
for (; i < aLen; i++) {
   if (a[i]%2 === 0){
      console.log(`${a[i]} is even.`);
   } else {
      console.log(`${a[i]} is odd.`);
   }
}
```

75

75

Video: For Loop



76

76

continue

```
a = [1,2,3,4,5,6,7,8,9];
let i = 0;
aLen = a.length;
for (; i < aLen; i++) {
   if (a[i]%2 === 0){
      console.log(`${a[i]} is even.`);
   } else {
      continue;
   }
}
```

77

77

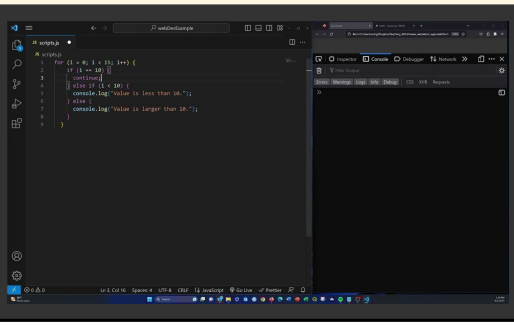
break

```
a = [1,2,3,4,5,6,7,8,9];
let i = 0;
aLen = a.length;
for (; i < aLen; i++) {
   if (a[i]%2 === 0){
    break;
   } else {
    console.log(`${a[i]} is odd.`);
   }
}
```

78

78

Video: Continue and Break



79

79

While Loop

Watch for infinite loops!!!!

```
let i = 20;
while (i > -1) {
  console.log("The number is " + i);
  i--;
}

let i = 0;
while (i < 20) {
  console.log("The number is " + i);
  i++;
}
```

80

80

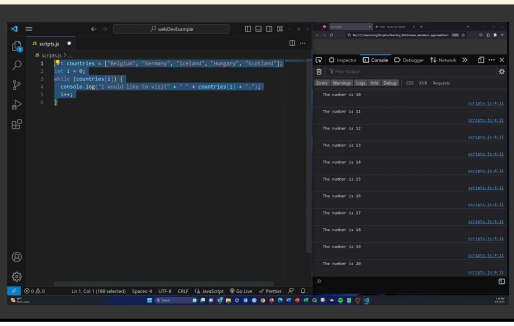
While Loop

```
let countries = ["Belgium", "Germany", "Iceland",
"Hungary", "Scotland"];
let i = 0;

while (countries[i]) {
  console.log("I would like to visit" + " " +
countries[i] + ".");
  i++;
}
```

81

81

Video: While Loop



82

82

Events

- Provides funtionality to webpages
- Allow webpages to accept input and respond or change in some way

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

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

83

83

GeoJSON

84

84

GeoJSON

- JSON = JavaScript Object Notion
- Used to store and exchange data
- Text written with JavaScript object notation
- GeoJSON = allows you to encode geographic structures
- Used in web applications
- One file can store multiple geometry types

http://geojson.org/

85

85

GeoJSON: Point


```
{
"type": "FeatureCollection",
"name": "json_point2",
"crs": { "type": "name", "properties": {
"name": "urn:ogc:def:crs:OGC:1.3:CRS84" }
},
"features": [
{ "type": "Feature", "properties": { "id": 1,
"attribute1": 1, "attribute2": "point1",
"attribute3": 0.1111 }, "geometry": { "type":
"Point", "coordinates": [ -96.437, 37.283 ] } }
]
}
```

86

86

GeoJSON: Point

```
{
"type": "FeatureCollection",
"name": "example_point",
"crs": { "type": "name", "properties": { "name":
"urn:ogc:def:crs:OGC:1.3:CRS84" } },
"features": [
{ "type": "Feature", "properties": { "id": 1, "attribute1": 1,
"attribute2": "point1", "attribute3": 0.1111 }, "geometry": { "type":
"Point", "coordinates": [ -96.437, 37.283 ] } },
{ "type": "Feature", "properties": { "id": 2, "attribute1": 2,
"attribute2": "point2", "attribute3": 0.22222 }, "geometry": {
"type": "Point", "coordinates": [ -83.006, 39.913 ] } },
{ "type": "Feature", "properties": { "id": 3, "attribute1": 3,
"attribute2": "point3", "attribute3": 0.33333 }, "geometry": {
"type": "Point", "coordinates": [ -115.666, 43.247 ] } },
{ "type": "Feature", "properties": { "id": null, "attribute1": null,
"attribute2": null, "attribute3": null }, "geometry": { "type":
"Point", "coordinates": [ -78.691, 40.501 ] } }
]
}
```


87

87

GeoJSON: Line


```
{
"type": "FeatureCollection",
"name": "line_example",
"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" } },
"features": [
{ "type": "Feature", "properties": { "id": 1, "attribute1": 1, "attribute2": 1111, "attribute3": "line" }, "geometry": {
"type": "MultiLineString", "coordinates": [ [ [ -118.903, 39.471 ], [ -112.086, 41.188 ], [ -105.957, 42.315 ], [ -
97.718, 41.237], [ -90.313, 39.030], [ -85.458, 36.234], [ -78.544, 38.883 ] ] ] } }
]
}
```

88

88

GeoJSON: Polygon

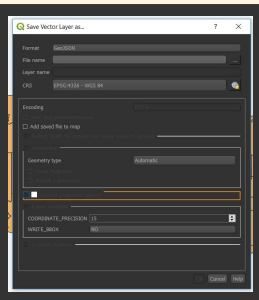

```
{
"type": "FeatureCollection",
"name": "polygon_example3",
"crs": { "type": "name", "properties": { "name":
"urn:ogc:def:crs:OGC:1.3:CRS84" } },
"features": [
{ "type": "Feature", "properties": { "id": 1 },
"geometry": { "type": "MultiPolygon",
"coordinates": [ [ [ [ -101.2, 43.051 ], [ -99.238,
45.111 ], [ -99.238, 45.111 ], [ -93.697, 46.239 ],
[ -89.332, 42.463 ], [ -88.254, 36.333 ], [ -
92.275, 31.429 ], [ -100.219, 32.312 ], [ -
107.183, 36.235 ], [ -105.809, 42.07 ], [ -101.2,
43.051 ] ], [ [ -99.581, 40.354 ], [ -98.503,
34.371 ], [ -93.501, 36.823 ], [ -92.128, 39.373
], [ -95.364, 41.923 ], [ -99.581, 40.354 ] ] ] ] }
}
]
}
```

89

89

GeoJSON in QGIS

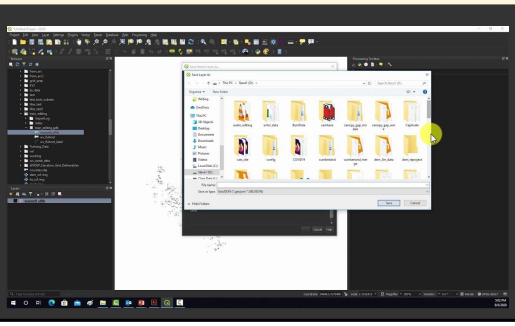
- Can export files to GeoJSON format
- Can create bounding box
- Can set coordinate precision
- QGIS can read and write many file types



90

90

Video: GeoJSON with QGIS



91

91

This is the end of this lecture module.

Please return to the West Virginia View Webpage for additional content.

92