Think: instead of functions but data so which container contaisn what varables? Think variables grouped

Think: Objects → Method/properties

History of JavaScript:

- Developed by netscape was caliied livescape
- Joint dev with sun Microsoft system 1995
- ECMA-262 or ECMAscript become standard 262 of European computer manufactures
- Cilent-side technology
  - Code is downloaded to web browser and executed by client.
  - Can't do direct manipulation of resources on server side (access data in central database)
  - Web client has full access to original script source code so user have ability to read the original code
- Can be used for host environment (server-side) such as node.js and electron for desktop
  - Global object is ← global scope for node.js (node. JavaScript)
  - Window object ← global scope for web browser environment
  - **Think node.js (**think Is terminal compiler)
- Console object ← display program output error messages (think JavaScript console in view developer chrome)

Javascript execution environment:

- Uses node.JS environment and the global object is: Global
- Object: Global contains: Properties + Methods
  - EG: Array, object, string math and properties specific for node.js
- For Object: Global properties are automatically available everywhere in script without object reference (think of object: global as before the main in c programming)
  - EG: console.log(global.parseFloat("3.14xradius");
    - Global is the object and parsefloat is method but since the object: global is available anywhere no need to include

Javascript web execution environment: (think html)

- Top level referencing environment for scripts
- Object: windows represent window in which the browser displays the document
- Object: window contains: values, function, constructors, objects (defined in javascript core)
- Variable and functions you declare are properties of object: windows

Console output:

- Display program output, error messages and other information
- Found in object: windows (web browser environment) and global (node.js)

User input:

- Found in object: windows (web browser environment) and global (node.js) but we will talk about global (node.js) syntax
- Readline-syn module
    - EG:

JavaScript primitive data types: JavaScript do not specify variable data type it is inferred during run time (c programming we normally state)

- Number type: a number (**Think in c programming we don't have float or integer or short to distinguish but by default stored as **a double precision position floating point values**)
- String: sequence of characters
- Boolean: true (any number 1,2,3) or false (0 value or empty string)
- Null: not pointing to anything or any object
    - Variable type is object
    - Output: 'null'

Objects:

- The purpose of objects is a computer representation of real objects in real world. Because in a real world we have many object such as: dogs, table, lights.
- **Objects** made up of →
    - **Properties** (information about particular object or set of variables)
    - **Behaviour** (things that the object can do or manipulate the data stored in object) represented through
        - *Methods* (or *functions* in c programming)

Type of object:

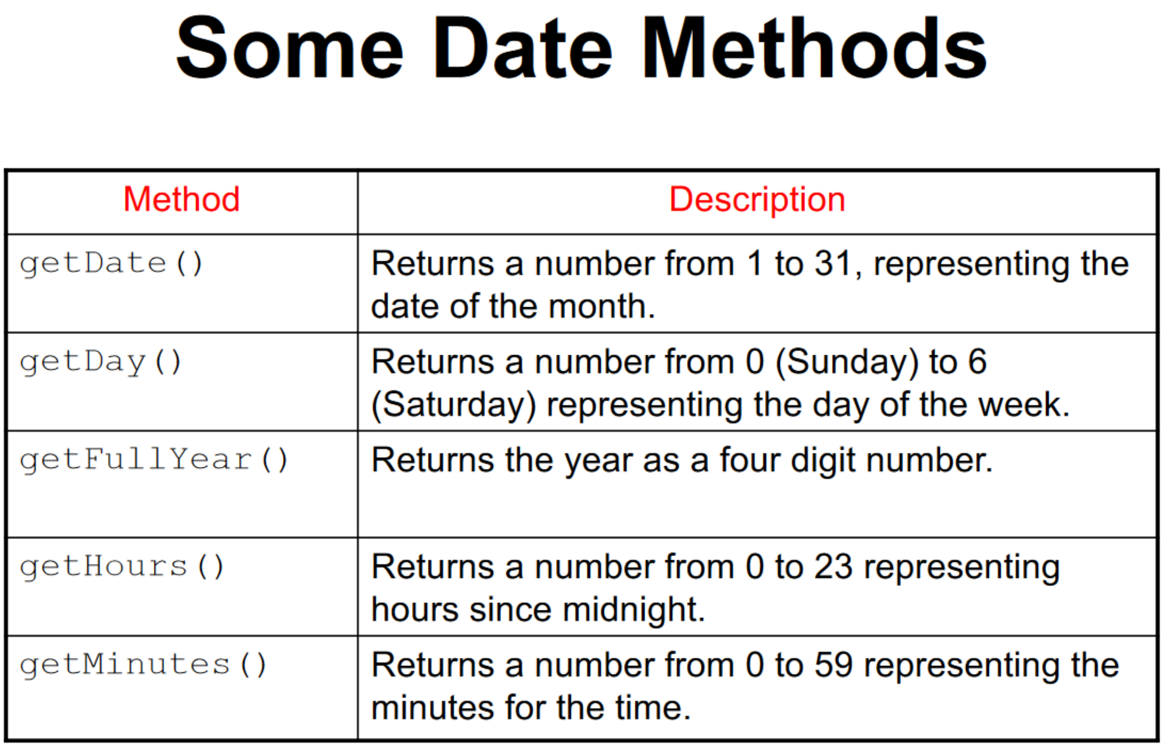
- Javascript core built-in objects: (think node.js)
    - They represent the data type:
        - Number, string, Boolean (once you declare these they become primitives data type as above), Array
    - For special task (think object constructor. Refer to bottom of sheet highlight for use)
        - Date, math, regexp, object, string

- Standard objects provided (built in) by web browser environment
    - They represent objects associated with web browser:
        - Navigator, window, history, location (currently url of window)

- HTML Document object model (DOM):
    - When web page is loaded the HTML page is represented as a tree of objects each object represent element (HTML, HEAD, BODY ELEMENT etc)
        - Document object
    - Purpose of JavaScript code is to manipulate document object model (tree)

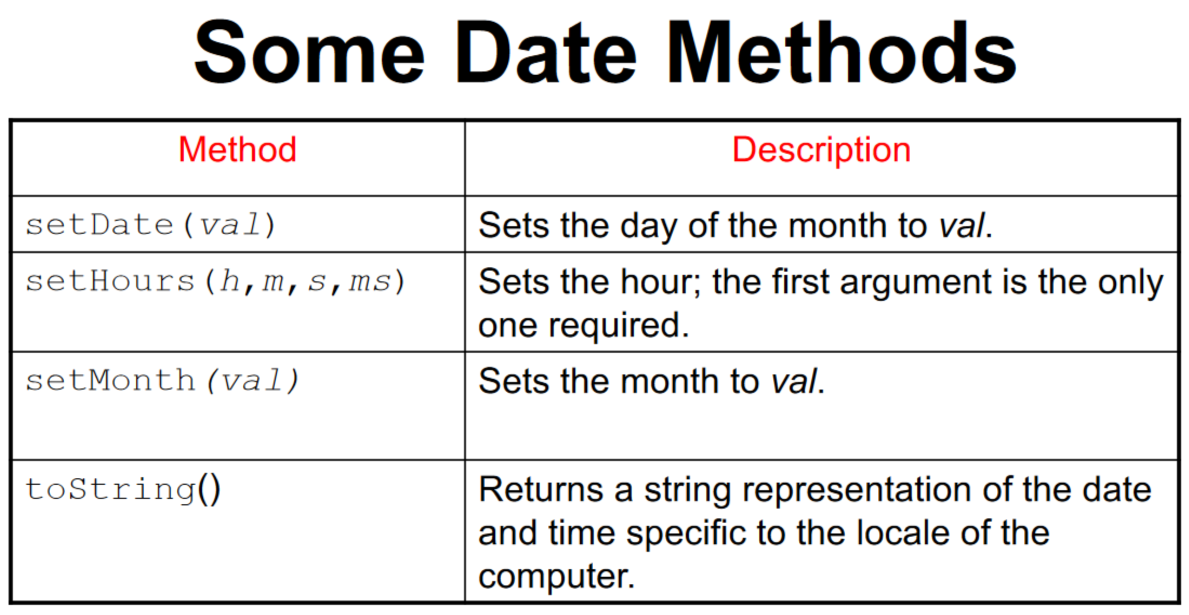
Object (constructor)**: Date** (refer to above)

- Methods (called behaviours ie things object can do) allow us to create and manipulate dates



Some Date Methods

Method	Description
`getDate()`	Returns a number from 1 to 31, representing the date of the month.
`getDay()`	Returns a number from 0 (Sunday) to 6 (Saturday) representing the day of the week.
`getFullYear()`	Returns the year as a four digit number.
`getHours()`	Returns a number from 0 to 23 representing hours since midnight.
`getMinutes()`	Returns a number from 0 to 59 representing the minutes for the time.

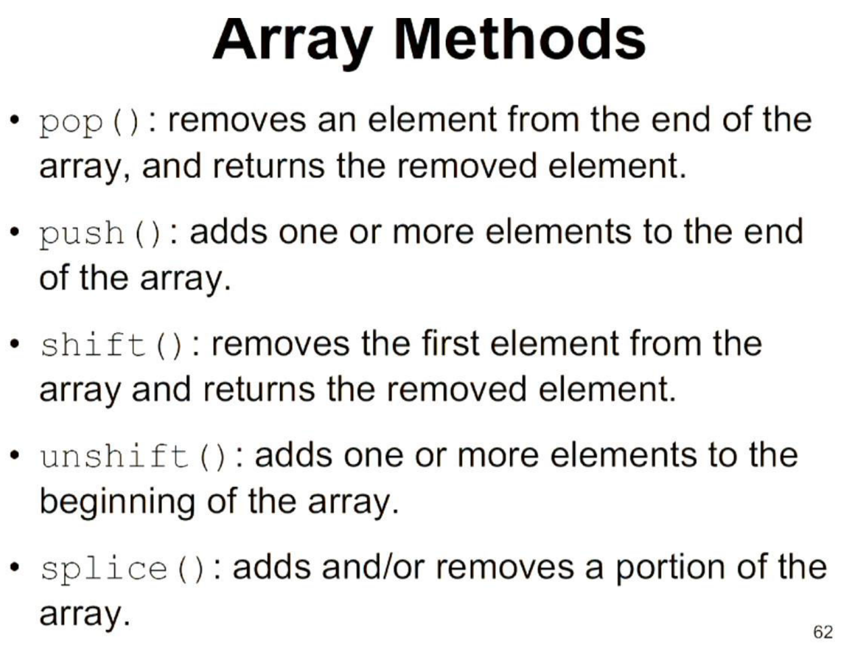


Some Date Methods

Method	Description
`setDate(val)`	Sets the day of the month to *val*.
`setHours(h,m,s,ms)`	Sets the hour; the first argument is the only one required.
`setMonth(val)`	Sets the month to *val*.
`toString()`	Returns a string representation of the date and time specific to the locale of the computer.

Object (constructor)**: Array** (refer to above)

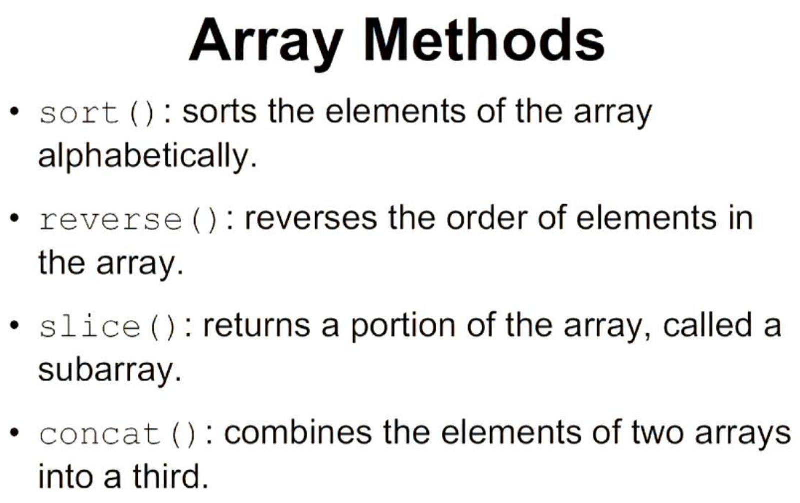
- List of variables that are usually related in some way and can be referenced using index
- The elements of a single array can contain: Numbers and strings doesn't have to be same type (**think in c programming an array had to be same type like number)
- To create one refer below for the **New and an object constructor method or second method** because its an object constructor
    - Syntax the array(array length/elements) ← That's the difference inside bracket array length OR
    - Var ArrayName = [element1, element2, element3..];
- Methods (behaviours)



Array Methods

- `pop()`: removes an element from the end of the array, and returns the removed element.
- `push()`: adds one or more elements to the end of the array.
- `shift()`: removes the first element from the array and returns the removed element.
- `unshift()`: adds one or more elements to the beginning of the array.
- `splice()`: adds and/or removes a portion of the array.

62

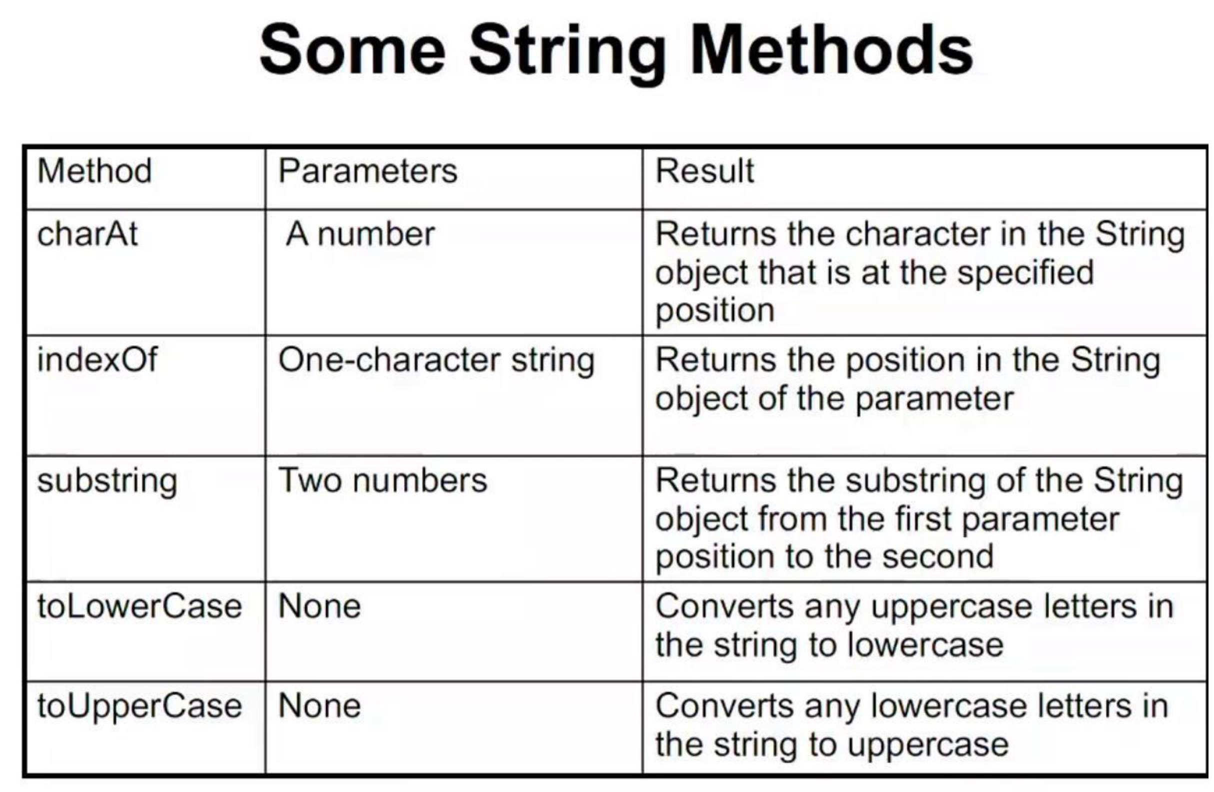


Array Methods

- `sort()`: sorts the elements of the array alphabetically.
- `reverse()`: reverses the order of elements in the array.
- `slice()`: returns a portion of the array, called a subarray.
- `concat()`: combines the elements of two arrays into a third.

Object (constructor)**: String** (refer to above)

- A lot of methods that allow you to manipulate strings



Some String Methods

Method	Parameters	Result
charAt	A number	Returns the character in the String object that is at the specified position
indexOf	One-character string	Returns the position in the String object of the parameter
substring	Two numbers	Returns the substring of the String object from the first parameter position to the second
toLowerCase	None	Converts any uppercase letters in the string to lowercase
toUpperCase	None	Converts any lowercase letters in the string to uppercase

- For: charAt indexOf, substring refer to powerpoint for detailed examples

Object in real world: EG: Car

- Properties: (what makes up the car)
    - The number of wheels
    - Height of car
    - Number of doors it has
- Behaviour:
    - Car make noise
    - Drive the car
    - Change the car colour

Notice: JavaScript has no classes only functions + objects

Var <Variable name> = <value>;

- Value: Could be string → 'John'
- Variable and function declaration is treated as if they're moved to the top of current scope
    - Eg: var bot
        - Output is 'undefined' but it is declared which is works
    - But the assignment of 'Value' isn't

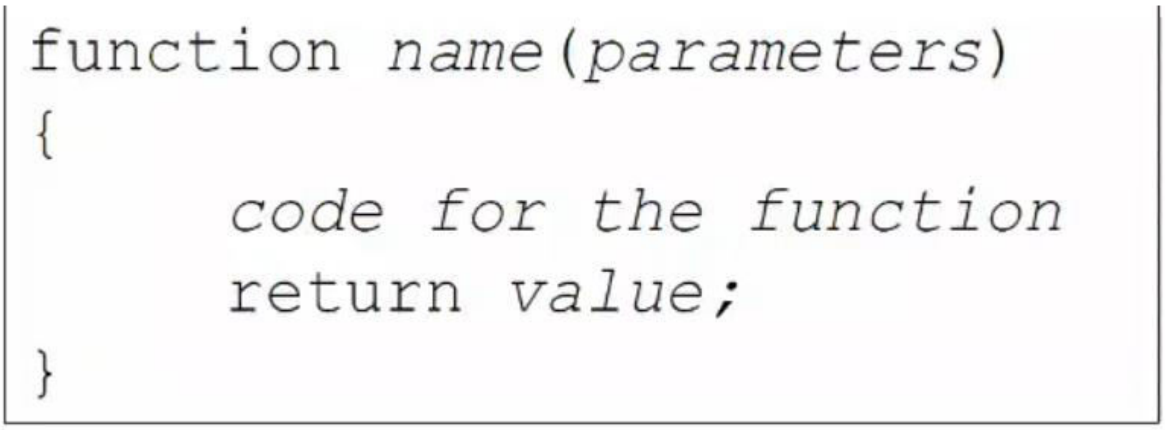
String → number/float/int (object: number)

- Console.log(Number("313") + 10)

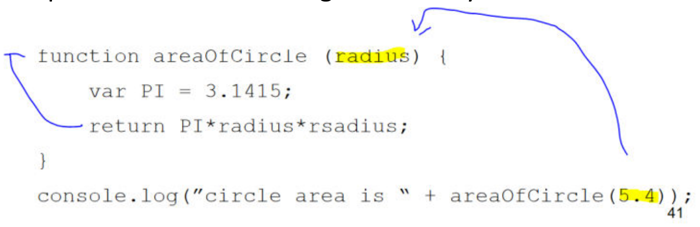
- Console.log(parsefloat/parseint("313=var) + 10)
    - Note: the bracket can also contain strings because it separates the numbers from other stuff
    - EG: console.log(parseint("50StringisHere")+10)
        - So the string "stringishere" is not converted

Returning single value from function:

Var



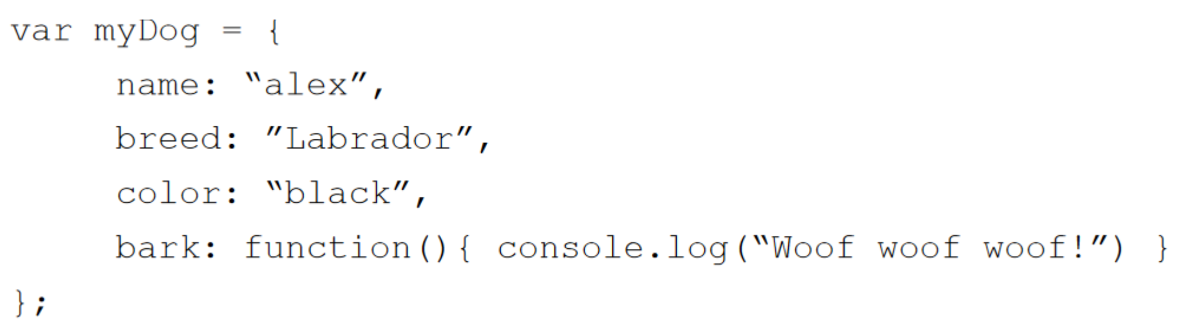
```
function name(parameters)
{
        code for the function
        return value;
}
```

- Return single value. For multiple values manipulate parameters.
- Use Var before function → var function(parameter)
- Don't say returning value data type (In c programming it would be int functioName) or parameter data type. Just use the variable name.
- Usually the parameters are values given already in main section of code ie:
    - 

```
function areaOfCircle (radius) {
    var PI = 3.1415;
    return PI*radius*rsadius;
}
console.log("circle area is " + areaOfCircle(5.4));
```

Creating object:

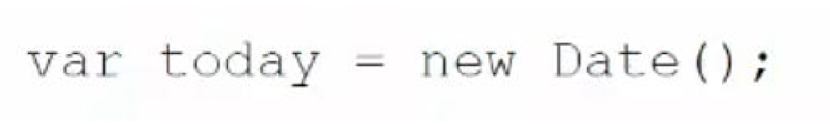
```
Var ObjectName = {

        //Properties

        Name: Value (or string),

        Name: Value,
        //Behaviours

        Name: function() {        what It does        };
```



```
var myDog = {
      name: "alex",
      breed: "Labrador",
      color: "black",
      bark: function(){ console.log("Woof woof woof!") }
};
```

OR New and an object constructor method (Think create new object based off the base of a built in object)

```
Var objectName = new objectConstructorName()
```



```
var today = new Date();
```

Accessing objects:

```
Var ObjectName = {

        //Properties

        Name: Value '(or string)',

        Name: Value,

        //Behaviours

        Name: method/function() {        what It does        };
```

→ Accessing the object

```
Var storage = Objectname.propertyName
Var storage = ObjectName.behaviourName() ← If it is accessing behaviours
```

OR

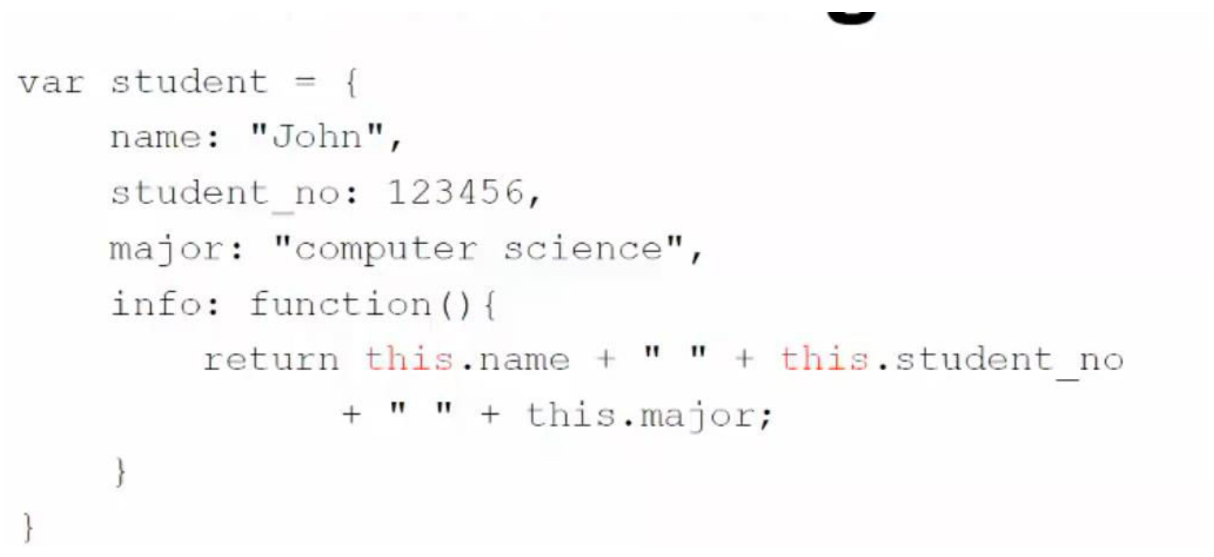
```
Var objectName = new object()
```

→ Accessing the object

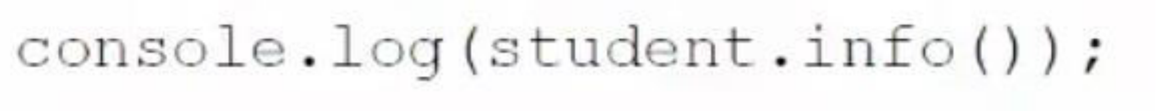
```
Var storage = objectName.methodName()
```

- So the methodName() are the in-built methods/functions of the object that has been referenced. (**Think we are creating new object based off a in-built object. The new object will have method/function of the in-built object)

Unique Behaviours



```
var student = {
     name: "John",
     student_no: 123456,
     major: "computer science",
     info: function(){
          return this.name + " " + this.student_no
               + " " + this.major;
     }
}
```



```
console.log(student.info());
```

- This is a keyword allows us to reference variables outside function