**JavaScript:**

**What is this?**

In JavaScript, the this keyword refers to an **object**. **Which** object depends on how this is being invoked (used or called). The this keyword refers to different objects depending on how it is used.

**Escape Character:**

|  |  |  |
| --- | --- | --- |
| \' | ' | Single quote |
| \" | " | Double quote |
| \\ | \ | Backslash |

**JS String Methods:**

**slice():** Extracts a part of a string and returns the extracted part in a new string. The method takes 2 parameters: the start position, and the end position (end not included).

**substring():** The difference is that start and end values less than 0 are treated as 0 in substring().

**substr():** The difference is that the second parameter specifies the **length** of the extracted part.

**replace():** By default, the replace() method is case sensitive. Writing MICROSOFT (with upper-case) will not work:

To replace case insensitive, use a **regular expression** with an /i flag (insensitive):

let text = "Please visit Microsoft!";  
let newText = text.replace(/MICROSOFT/i, "W3Schools");

To replace all matches, use a **regular expression** with a /g flag (global match):

**concat():** Joins two or more strings . The concat() method can be used instead of the plus operator. These two lines do the same:

let text1 = "Hello";  
let text2 = "World";  
let text3 = text1.concat(" ", text2);

**trim():** Removes whitespace from both sides of a string:

**padStart():** Method pads a string with another string:

## let text = "5"; Output: xxx5 let padded = text.padStart(4,"x");

**match():** The match() method searches a string for a match against a regular expression, and returns the matches, as an Array object.

let text = "The rain in SPAIN stays mainly in the plain";  
text.match(/ain/g);

**includes():** The includes() method returns true if a string contains a specified value.

**JS Number Methods:**

**toExponential():** returns a string, with a number rounded and written using exponential notation.A parameter defines the number of characters behind the decimal point:

**toFixed():**  returns a string, with the number written with a specified number of decimals:

**toPrecision():** returns a string, with a number written with a specified length:

|  |  |
| --- | --- |
| **Method** | **Description** |
| Number() | Returns a number, converted from its argument. |
| parseFloat() | Parses its argument and returns a floating point number |
| parseInt() | Parses its argument and returns an integer |

**JS Array Methods:**

**join():** The join() method also joins all array elements into a string.It behaves just like toString(), but in addition you can specify the separator:

**shift():** The shift() method removes the first array element and "shifts" all other elements to a lower index.

**unshift():** The unshift() method adds a new element to an array (at the beginning), and "unshifts" older elements:

**splice():** The splice() method can be used to add new items to an array. The first parameter (2) defines the position **where** new elements should be **added** (spliced in).The second parameter (0) defines **how many** elements should be **removed**.

**sort():**

const points = [40, 100, 1, 5, 25, 10];  
points.sort(function(a, b){return a - b});

const points = [40, 100, 1, 5, 25, 10];  
points.sort(function(a, b){return b - a});

**JS Loop For In:** The JavaScript for in statement loops through the properties of an Object: The JavaScript for in statement can also loop over the properties of an Array:

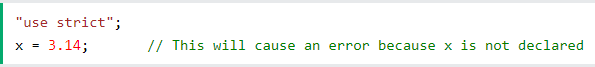
**JS Loop For Of:** The JavaScript for of statement loops through the values of an iterable object.It lets you loop over iterable data structures such as Arrays, Strings, Maps, NodeLists, and more:

## JavaScript Objects vs Maps:

|  |  |  |
| --- | --- | --- |
| **Iterable** | Not directly iterable | Directly iterable |
| **Size** | Do not have a size property | Have a size property |
| **Key Types** | Keys must be Strings (or Symbols) | Keys can be any datatype |
| **Key Order** | Keys are not well ordered | Keys are ordered by insertion |
| **Defaults** | Have default keys | Do not have default keys |

## JS Strict Mode:

## “use strict"; Defines that JavaScript code should be executed in "strict mode". The "use strict" Directive The "use strict" directive was new in ECMAScript version 5.It is not a statement, but a literal expression, ignored by earlier versions of JavaScript. The purpose of "use strict" is to indicate that the code should be executed in "strict mode". With strict mode, you can not, for example, use undeclared variables. All modern browsers support "use strict" except Internet Explorer 9 and lower:



**JS Versions:**

|  |  |  |
| --- | --- | --- |
| **Ver** | **Official Name** | **Description** |
| ES1 | ECMAScript 1 (1997) | First edition |
| ES2 | ECMAScript 2 (1998) | Editorial changes |
| ES3 | ECMAScript 3 (1999) | Added regular expressions Added try/catch Added switch Added do-while |
| ES4 | ECMAScript 4 | Never released |
| ES5 | ECMAScript 5 (2009)  [Read More](https://www.w3schools.com/js/js_es5.asp) | Added "strict mode" Added JSON support Added String.trim() Added Array.isArray() Added Array iteration methods Allows trailing commas for object literals |
| ES6 | ECMAScript 2015  [Read More](https://www.w3schools.com/js/js_es6.asp) | Added let and const Added default parameter values Added Array.find() Added Array.findIndex() |
|  | ECMAScript 2016  [Read More](https://www.w3schools.com/js/js_2016.asp) | Added exponential operator (\*\*) Added Array.includes() |
|  | ECMAScript 2017  [Read More](https://www.w3schools.com/js/js_2017.asp) | Added string padding Added Object.entries() Added Object.values() Added async functions Added shared memory |
|  | ECMAScript 2018  [Read More](https://www.w3schools.com/js/js_2018.asp) | Added rest / spread properties Added asynchronous iteration Added Promise.finally() Additions to RegExp |

**JS History:**

**JavaScript** was invented by **Brendan Eich** in 1995.It was developed for **Netscape 2**, and became the **ECMA-262** standard in 1997.After Netscape handed JavaScript over to ECMA, the Mozilla foundation continued to develop JavaScript for the Firefox browser. Mozilla's latest version was 1.8.5. (Identical to ES5).**Internet Explorer** (IE4) was the first browser to support ECMA-262 Edition 1 (ES1).

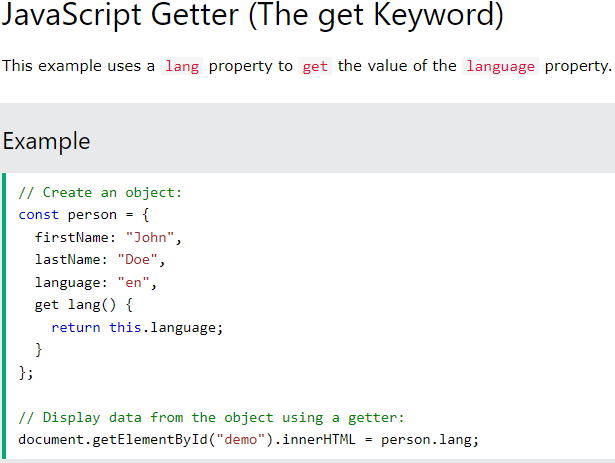
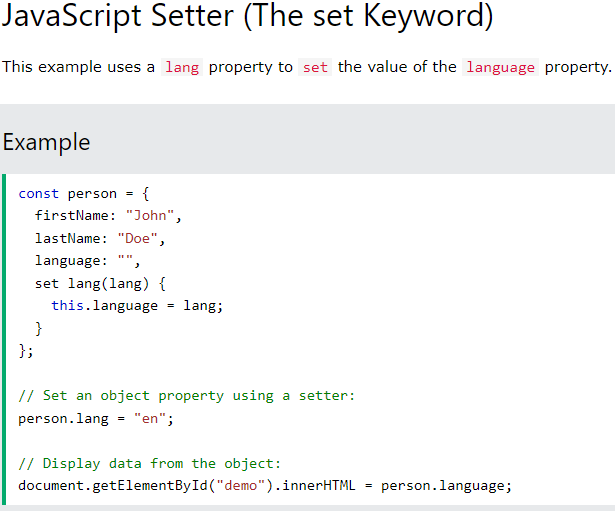
**JS Objects:**

## *Object Display:*

## Using JSON.stringify():

## Any JavaScript object can be stringified (converted to a string) with the JavaScript function JSON.stringify():

# ***Object Accessors:***

******

## Object.defineProperty():

## The Object.defineProperty() method can also be used to add Getters and Setters:

The Symbol.iterator is a function that returns a next() function. An iterable can be iterated over with the code: for (const x of iterable) { }

# ***Object Reference:***

### **Managing Objects**

// Create object with an existing object as prototype  
Object.create()  
  
// Adding or changing an object property  
Object.defineProperty(object, property, descriptor)  
  
// Adding or changing object properties  
Object.defineProperties(object, descriptors)  
  
// Accessing Properties  
Object.getOwnPropertyDescriptor(object, property)  
  
// Returns all properties as an array  
Object.getOwnPropertyNames(object)  
  
// Accessing the prototype  
Object.getPrototypeOf(object)  
  
// Returns enumerable properties as an array  
Object.keys(object)

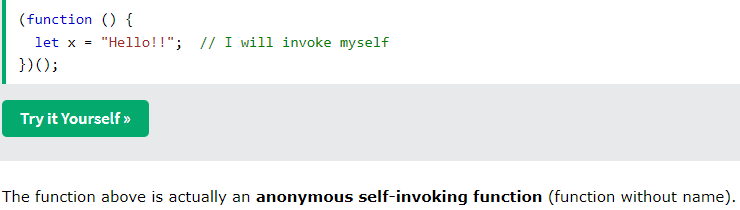
### **Protecting Objects**

// Prevents adding properties to an object  
Object.preventExtensions(object)  
  
// Returns true if properties can be added to an object  
Object.isExtensible(object)  
  
// Prevents changes of object properties (not values)  
Object.seal(object)  
  
// Returns true if object is sealed  
Object.isSealed(object)  
  
// Prevents any changes to an object  
Object.freeze(object)  
  
// Returns true if object is frozen  
Object.isFrozen(object)

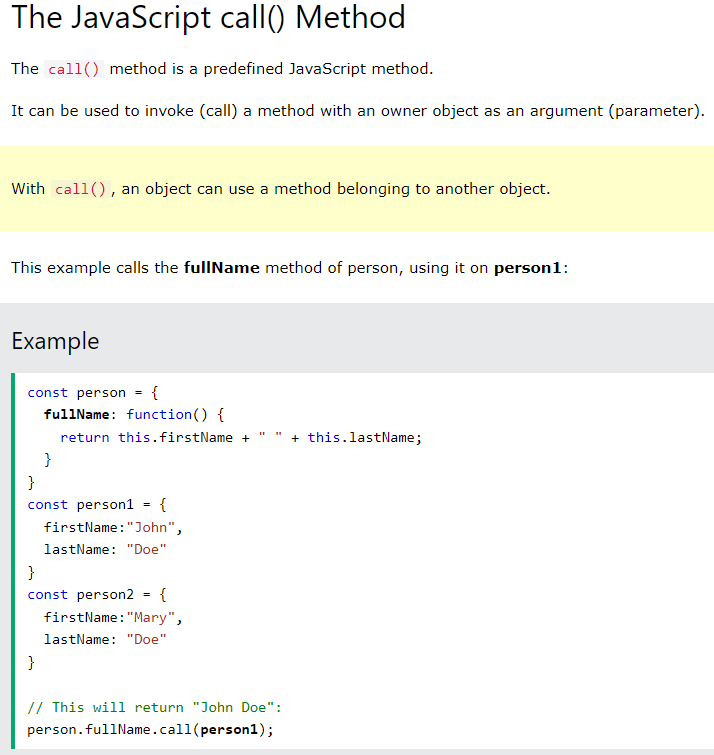
**JS Functions:**

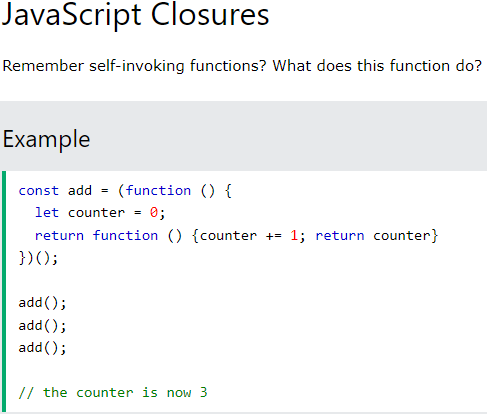
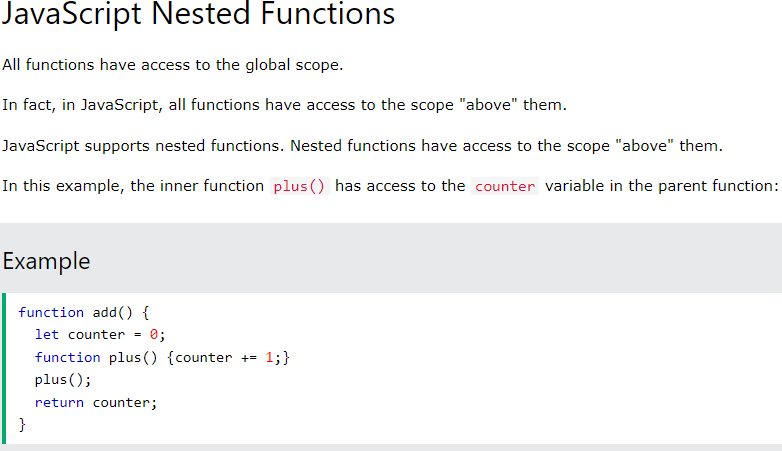
# ***Function Definitions:***

## Self-Invoking Functions Function expressions can be made "self-invoking". A self-invoking expression is invoked (started) automatically, without being called. Function expressions will execute automatically if the expression is followed by ().You cannot self-invoke a function declaration. You have to add parentheses around the function to indicate that it is a function expression:

****

# ***Function Call: Function Apply:***

******

***Function Closures:***