Frontend Development II

Functions Methods & Objects

Browsers require very detailed instructions about what we want them to do. Therefore, complex scripts can run to hundreds (even thousands) of lines. Programmers use functions, methods, and objects to organize their code.

Functions & Methods

Functions consist of a series of statements that have been grouped together because they perform a specifictask. A method is the same as a function, except methods are created inside (and are part of an object.

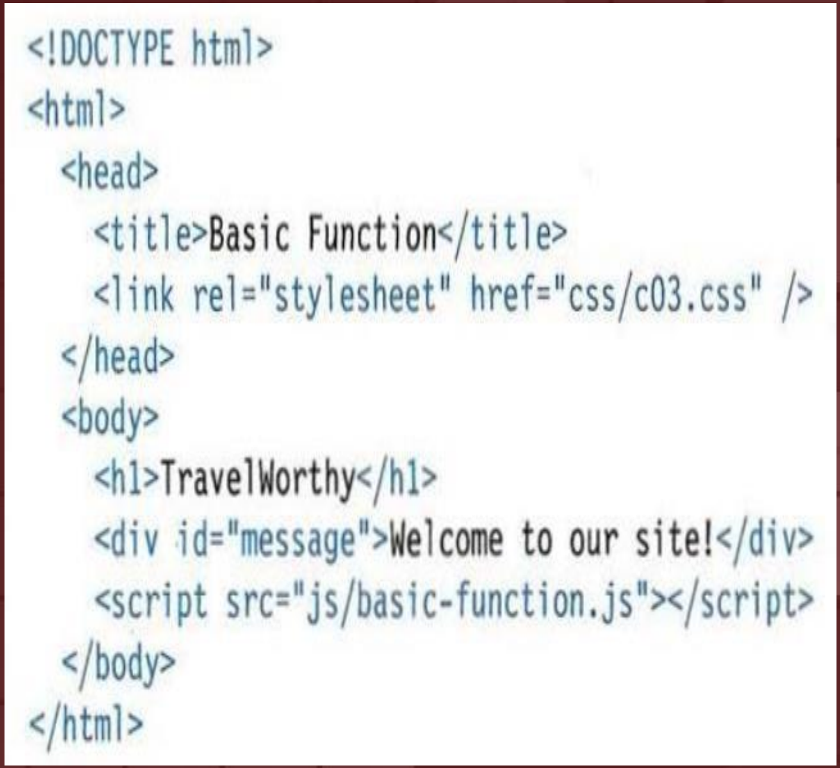
Functions :

Functions let you group a series of statements together to perform a specific task. If different parts of a script repeat the same task, you can reuse the function (rather than repeating the same se t of statements).

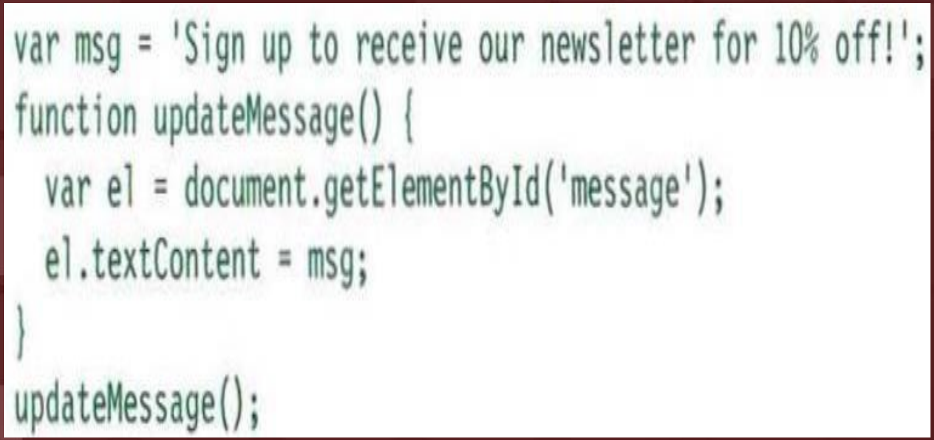
If you are going to ask the function to perform its task later, you need to give your function a name. That name should describe the task it is performing. When you ask it to perform its task, it is known as calling the function.

Some functions needs to be provided with information in order to achieve a given task. For example a function to calculate the area of a box would needs to know its width and height. Pieces of information passed to a function are known **parameters**

When you write a function and you expect it to provide you with an answer, the response is known as **return value**



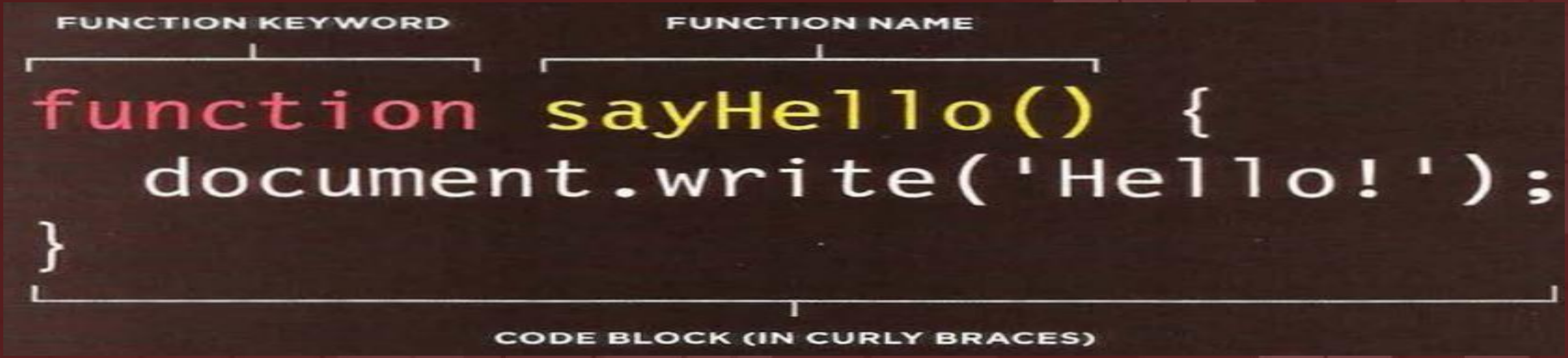
```
<!DOCTYPE html>
<html>
  <head>
    <title>Basic Function</title>
    <link rel="stylesheet" href="css/c03.css" />
  </head>
  <body>
    <h1>TravelWorthy</h1>
    <div id="message">Welcome to our site!</div>
    <script src="js/basic-function.js"></script>
  </body>
</html>
```



```
var msg = 'Sign up to receive our newsletter for 10% off!';
function updateMessage() {
  var el = document.getElementById('message');
  el.textContent = msg;
}
updateMessage();
```

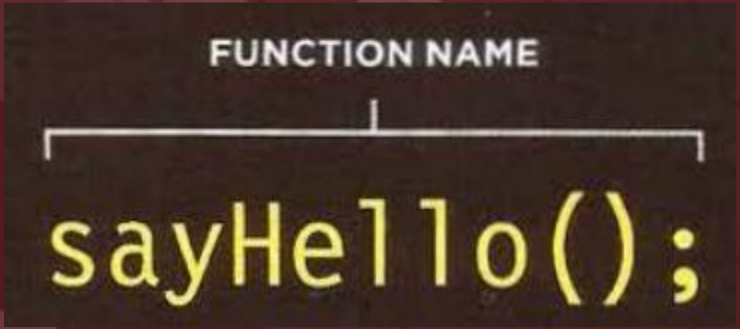
Declare a Function

You declare a function using declare keyword. You give the function a name followed by parenthesis. The statements that perform the task sit in a code block



```
function sayHello() {
    document.write('Hello!');
}
```

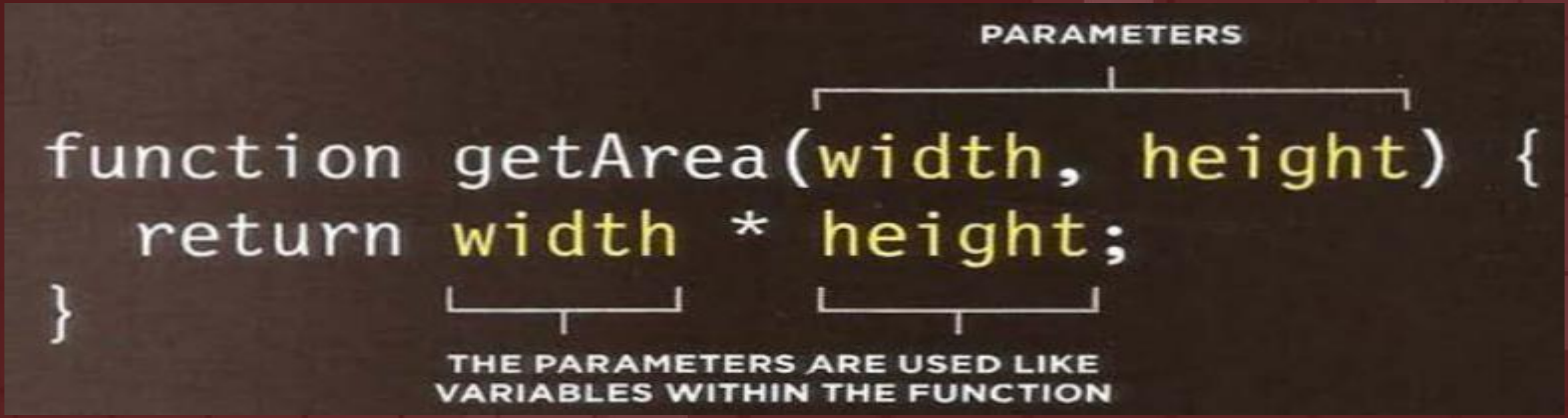
Having declared the function you can then execute all of the statements between its curly braces with just one line of code. This is known as calling the function



```
sayHello();
```

Parameters

Sometimes a function needs specific information to perform its task. In such cases , when you declare the function you give it parameters. Inside the function, the parameters act like variables

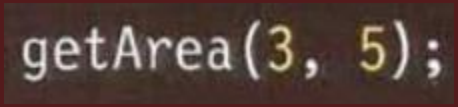


```
function getArea(width, height) {
   return width * height;
}
```

This function will calculate and return the area of a rectangle . To do this it needs the rectangle width and height. Each time you call the function theses values could bedifferent.

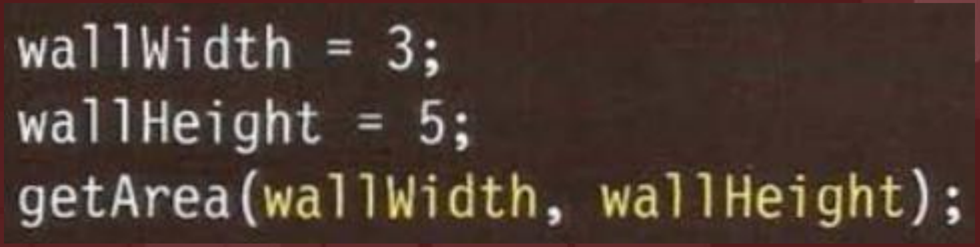
When you call a function that has parameters you specify the values it should use in the parenthesis that follow its name. The values are called parameters they can be provided as values or variables

Arguments as values



```
getArea(3, 5);
```

Arguments as variables



```
wallWidth = 3;
wallHeight = 5;
getArea(wallWidth, wallHeight);
```

VARIABLE SCOPE

The location where you declare a variable will affect where it can be used within your code. If you declare it within a function, it can only be used within that function. This is known as the variable's scope.

LOCAL VARIABLES

When a variable is created inside a function using the var keyword, it can only be used in that function.

It is called a local variable or function-level variable. It is said to have local scope or function-level scope. It cannot be accessed outside of the function in which it was declared.

GLOBAL VARIABLES

If you create a variable outside of a function, then it can be used anywhere within the script. It is called a global variable and has global scope.

```
function getArea(width, height){
var area = width * height;
return area;
}
var wallSize = getArea(3,2);
document. write(wallSize);
```

Which one is local/global?

WHAT IS AN OBJECT?

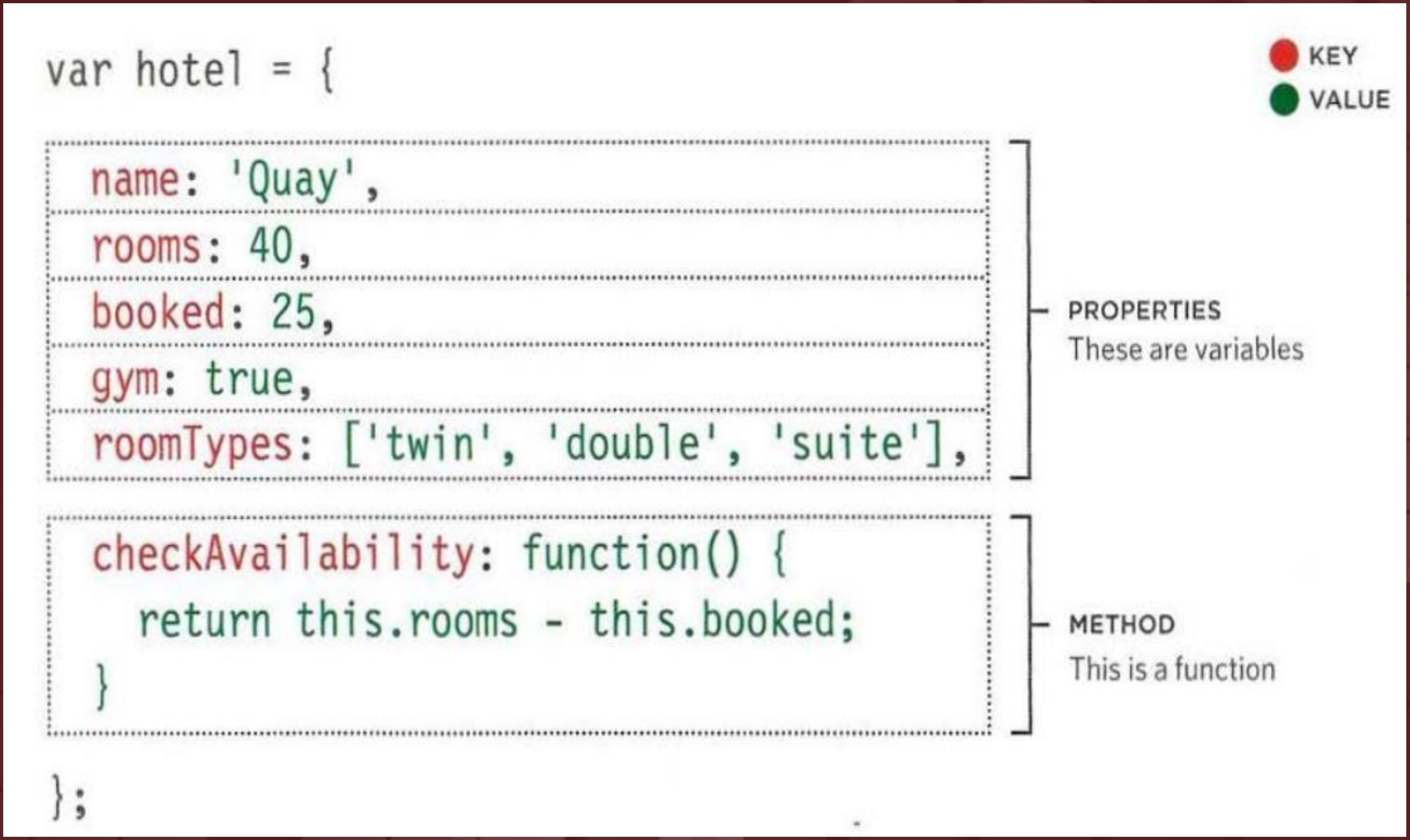
Objects group together a set of variables and functions to create a model of a something you would recognize from the real world. In an object, variables and functions take on new names.

IN AN OBJECT: VARIABLES BECOME KNOWN AS PROPERTIES

If a variable is part of an object, it is called a property . Properties tell us about the object, such as the name of a hotel or the number of rooms it has. Each individual hotel might have a different name and a different number of rooms.

IN AN OBJECT: FUNCTIONS BECOME KNOWN AS METHODS

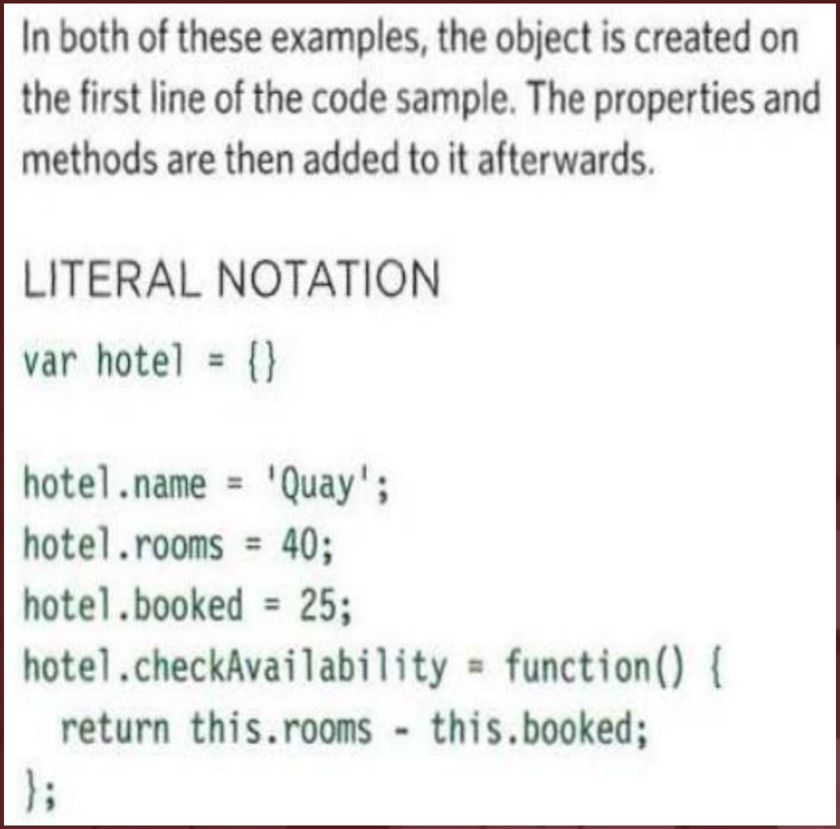
If a function is part of an object, it is called a method. Methods represent tasks that are associated with the object. For example, you can check how many rooms are available by subtracting the number of booked rooms from the tot al number of rooms.



```
var hotel = {

  name: 'Quay',
  rooms: 40,
  booked: 25,
  gym: true,
  roomTypes: ['twin', 'double', 'suite'],

  checkAvailability: function() {
    return this.rooms - this.booked;
  }

};
```

Like variables and named functions, properties and methods have a name and a value. In an object, that name is called a Key. The value of a property can be a string, number, Boolean, array, or even another object. The value of a method is always a function.

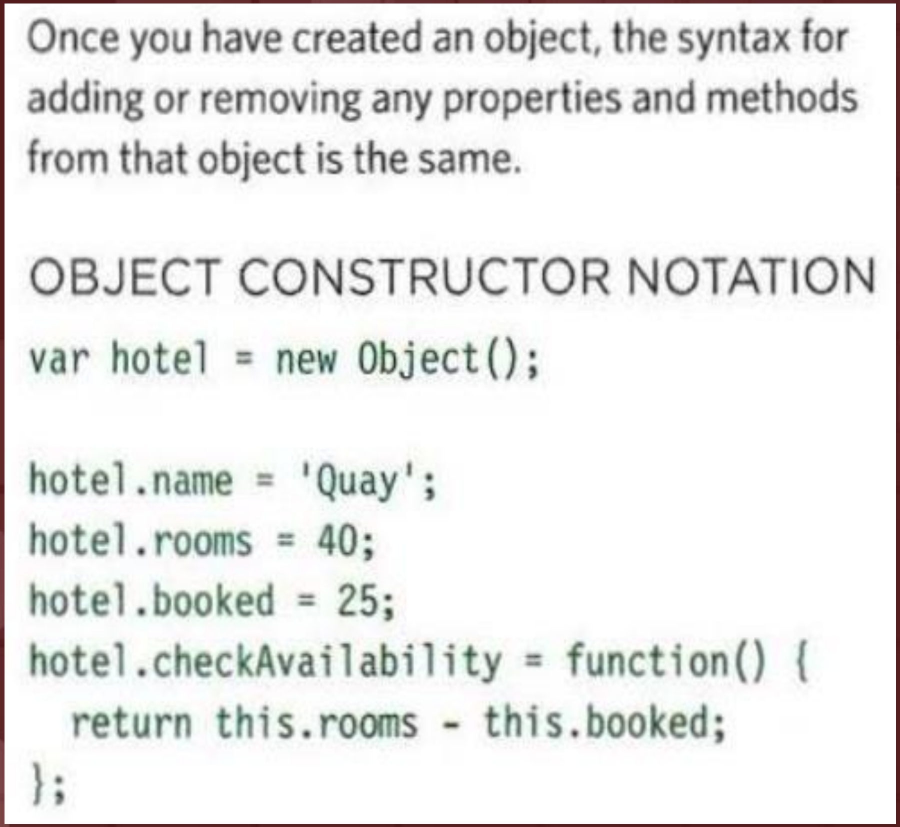
CREATE THE OBJECT, THEN ADD PROPERTIES & METHODS



In both of these examples, the object is created on the first line of the code sample. The properties and methods are then added to it afterwards.

LITERAL NOTATION

```
var hotel = {}

hotel.name = 'Quay';
hotel.rooms = 40;
hotel.booked = 25;
hotel.checkAvailability = function() {
  return this.rooms - this.booked;
};
```



Once you have created an object, the syntax for adding or removing any properties and methods from that object is the same.

OBJECT CONSTRUCTOR NOTATION

```
var hotel = new Object();

hotel.name = 'Quay';
hotel.rooms = 40;
hotel.booked = 25;
hotel.checkAvailability = function() {
  return this.rooms - this.booked;
};
```

CREATING AN OBJECT WITH PROPERTIES & METHODS

LITERAL NOTATION

A colonseparates the key/value pairs.
There is acommabetweencach ley/value pair.

```
var hotel = {
  name: 'Quay'
  rooms: 40,
  boofed: 25,
  ‹hecb vaitadility: function() (
    r0turn thi5.m0N5 - ttis,boOled;
  }
};
```

OBJECTC0NSTRUCT0RN0TAT1ON

The function can be used to create multiple objects.
The th1s keyword is used instead of tle objxt name.

```
function Fotel (naxe, nans, 9ooked) (
  this.nae = nae;
  th1s.ro0m5 ” roas;
  tfils.b00lfl4 • fioote4;
  this.checkAvai1ability • function() (
    return th1s.poxs - th1s.booked,
  };
}
var qvayEote1 • nex Hotel('§uay', 40, 2s);
var parfHoteJ - ne Hote1('Park', 120, 77);
```

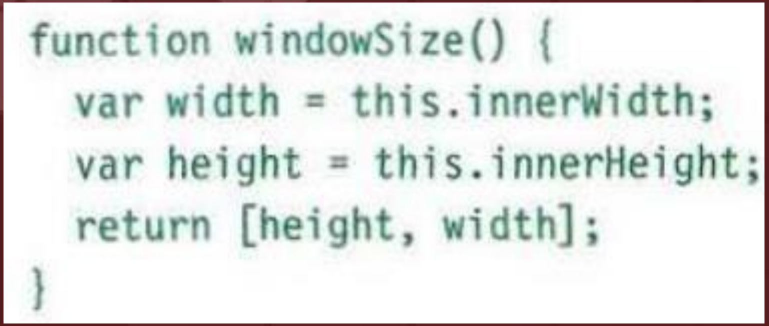
THIS (IT IS A KEYWORD)

The keyword this is commonly used inside functions and objects. Where the function is declared alters what this means. It always refers to one object, usually the object in which the function operates.

A FUNCTION IN GLOBAL SCOPE

When a function is created at the top level of a script (that is, not inside another object or function), then it is in the global scope or global context.

The default object in this context is the window object. so when this is used inside a function in the global context it refers to the window object. Below, this is being used to return properties of the window object

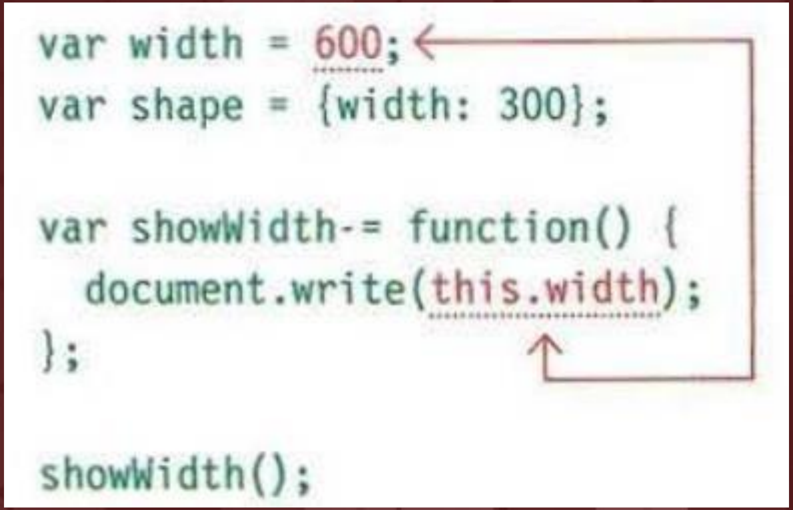


```
function windowSize() {
  var width = this.innerWidth;
  var height = this.innerHeight;
  return [height, width];
}
```

GLOBAL VARIABLES

All global variables also become properties of the window object. so when a function is in the global context, you can access global variables using the window object, as well as its other properties.

Here, the showWidth () function is in global scope, and this.width refers to the width variable:



```
var width = 600;
var shape = {width: 300};

var showWidth-= function() {
  document.write(this.width);
};

showWidth();
```

Here, the function would write a value of 600 into the page (using the document object's write() method).

fppt.com

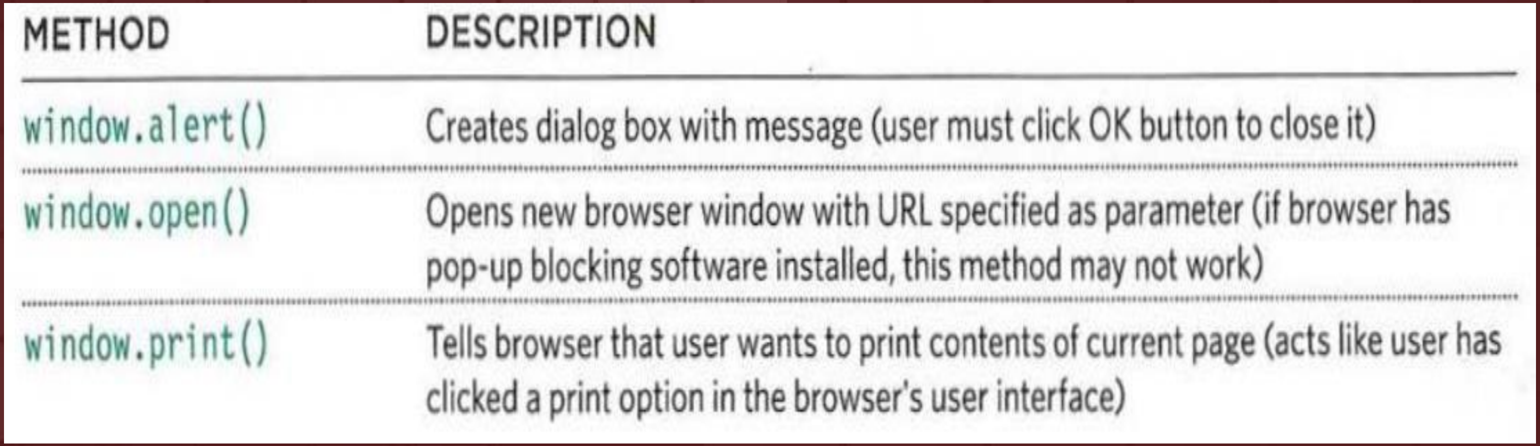
WHAT IS AN OBJECT MODEL?

You have seen that an object can be used to create a model of something from the real world using data. An object model is a group of objects, each of which represent related things from the real world. Together they form a model of something larger.

THE BROWSER OBJECT MODEL:

THE WINDOW OBJECT

The window object represents the current browser window or tab. It is the topmost object in the Browser Object Model, and it contains other objects that tell you about the browser.



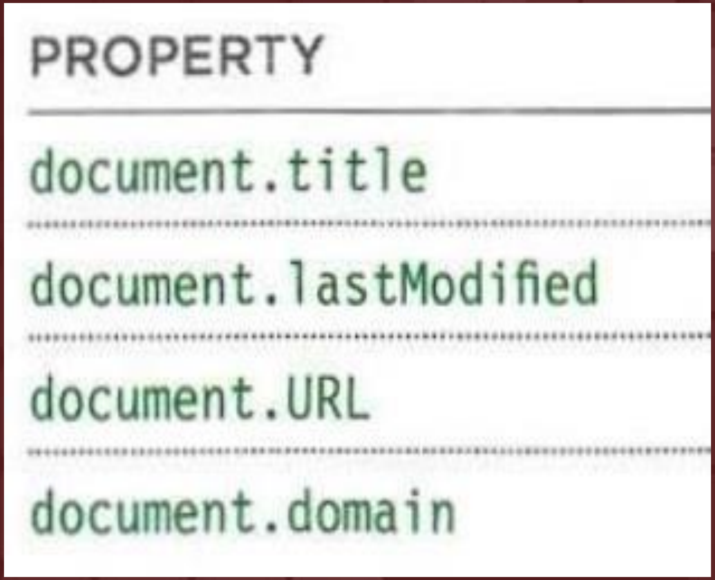
METHOD	DESCRIPTION
window.alert()	Creates dialog box with message (user must click OK button to close it)
window.open()	Opens new browser window with URL specified as parameter (if browser has pop-up blocking software installed, this method may not work)
window.print()	Tells browser that user wants to print contents of current page (acts like user has clicked a print option in the browser's user interface)

PROPERTY	DESCRIPTION
window. innerHe ight	Height of window (excluding browser chrome/user interface) (in pixels)
xindov.inMeWXi) h	Width of window (excluding browser chrome/user interface) (in pixelsJ
xindo•. pageX0ffset	Distance document has been scrolled horizontally (in pixels)
window. PageY0ffset	Distance document has been scrolled vertically (in pixels)
window. screenX	X-coordinate of potnter, relative to top left corner of screen (in pixels)
xindox. screenY	Y-coordinate of pointer, relative to top leH cofner ol screen (in pixels)
vi ndov.1ocati on	Current URL of xindoa object (or local file path)
xindo•.document	Reference to docullent object, which is used to represent the current page contained in window
xindox. history	Reference to history object for browser window or tab, which contains details of the pages that have been viewed in that window or tab
vlndov. h1story. Jength	Number of items in history object for browsef window or tab
xindox. screen	Reference to screen object
xindox. screen.xidth	Accesses screen object and finds value of its xidth property (in plxels)
xihdox. screen.height	Accesses screen object and finds value of its helght property fin pixels)

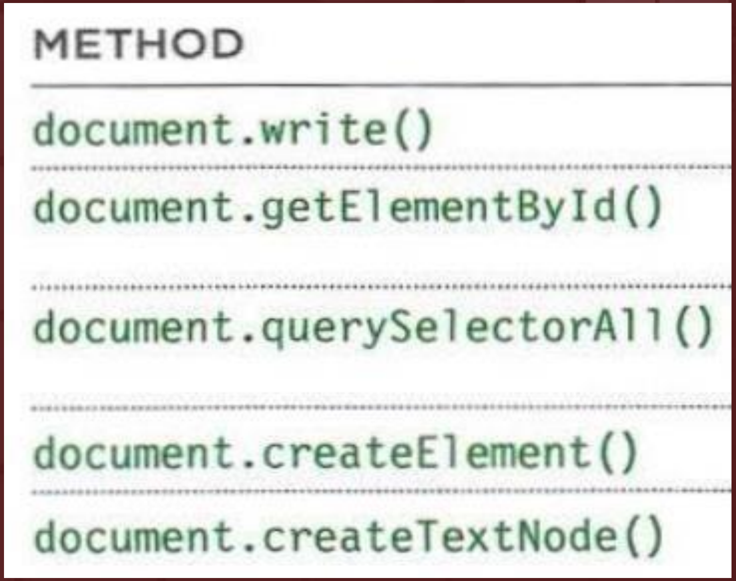
THE DOCUMENT OBJECT MODEL:
THE DOCUMENT OBJECT

The topmost object in the Document Object Model (or DOM) is the document object. It represents the web page loaded into the current browser window or tab.

Here are some properties and method of the document object, which tell you about the current page.



PROPERTY
document.title
document.lastModified
document.URL
document.domain

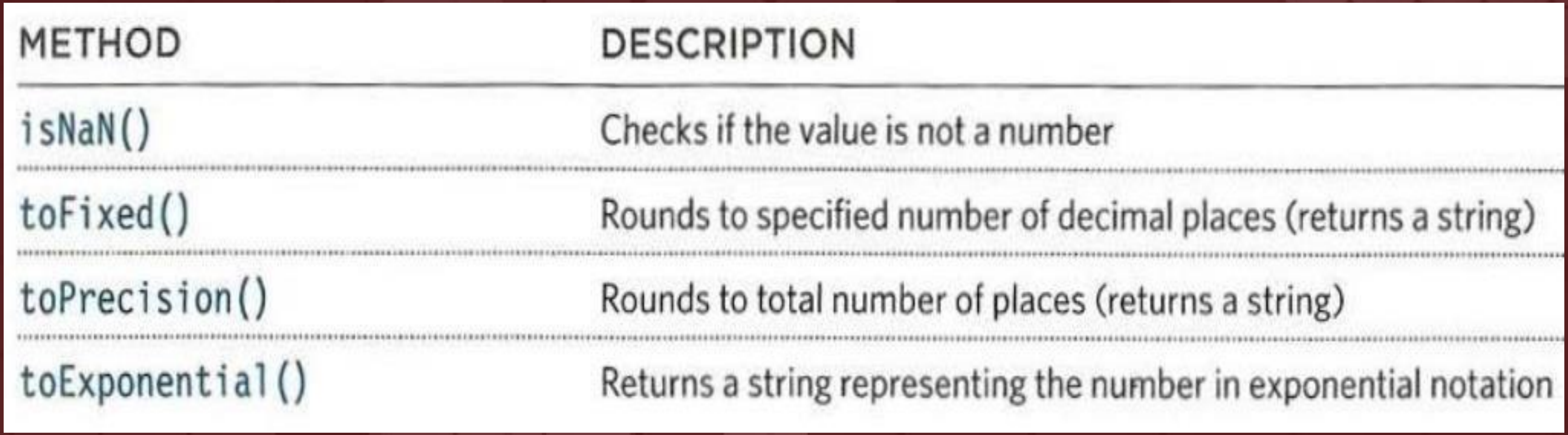


METHOD
document.write()
document.getElementById()
document.querySelectorAll()
document.createElement()
document.createTextNode()

GLOBAL OBJECTS:

NUMBER OBJECT

Whenever you have a value that is a number,you can use the methods and properties of the Number object on it.These methods are helpful when dealing with a range of applications from financial calculations to animations.

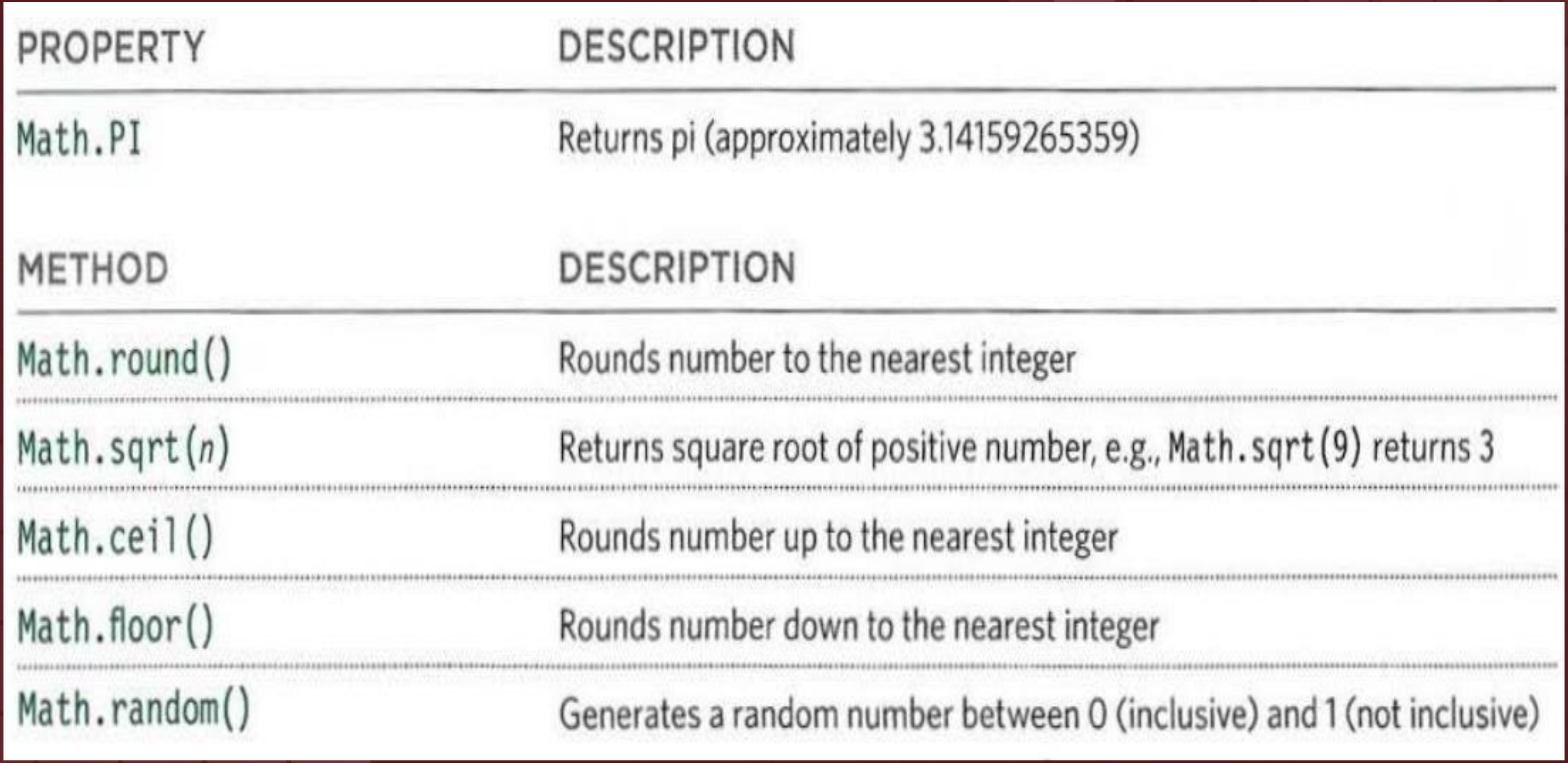


METHOD	DESCRIPTION
isNaN()	Checks if the value is not a number
toFixed()	Rounds to specified number of decimal places (returns a string)
toPrecision()	Rounds to total number of places (returns a string)
toExponential()	Returns a string representing the number in exponential notation

GLOBAL OBJECTS:

MATH OBJECT

The Math object has properties and methods for mathematical constants and functions.

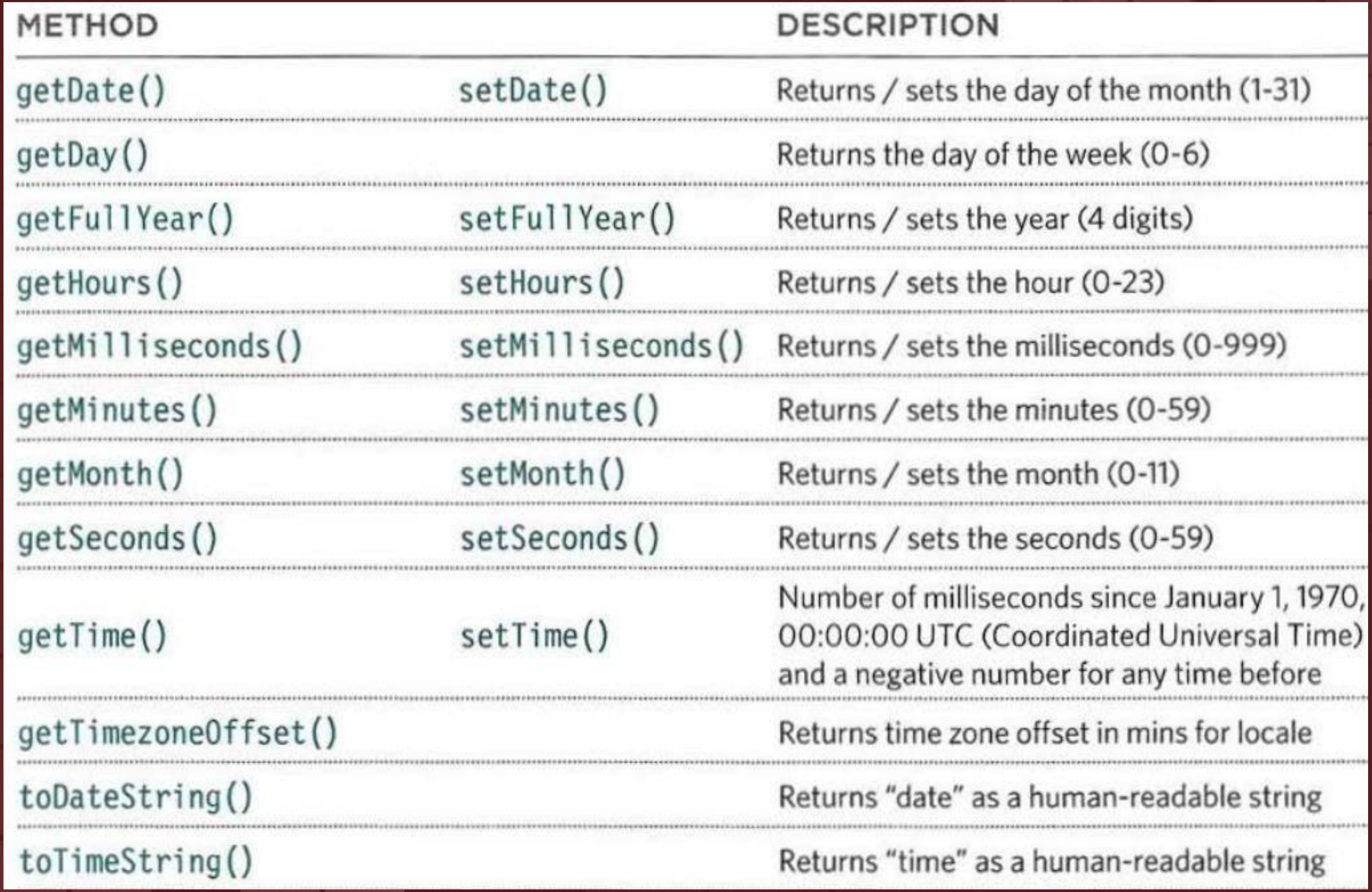


PROPERTY	DESCRIPTION
`Math.PI`	Returns pi (approximately 3.14159265359)

METHOD	DESCRIPTION
`Math.round()`	Rounds number to the nearest integer
`Math.sqrt(n)`	Returns square root of positive number, e.g., `Math.sqrt(9)` returns 3
`Math.ceil()`	Rounds number up to the nearest integer
`Math.floor()`	Rounds number down to the nearest integer
`Math.random()`	Generates a random number between 0 (inclusive) and 1 (not inclusive)

GLOBAL OBJECTS:
DATE OBJECT (AND TIME)

Once you have created a Date object, the following methods let you set and retrieve the time and date that it represents.



METHOD		DESCRIPTION
`getDate()`	`setDate()`	Returns / sets the day of the month (1-31)
`getDay()`		Returns the day of the week (0-6)
`getFullYear()`	`setFullYear()`	Returns / sets the year (4 digits)
`getHours()`	`setHours()`	Returns / sets the hour (0-23)
`getMilliseconds()`	`setMilliseconds()`	Returns / sets the milliseconds (0-999)
`getMinutes()`	`setMinutes()`	Returns / sets the minutes (0-59)
`getMonth()`	`setMonth()`	Returns / sets the month (0-11)
`getSeconds()`	`setSeconds()`	Returns / sets the seconds (0-59)
`getTime()`	`setTime()`	Number of milliseconds since January 1, 1970, 00:00:00 UTC (Coordinated Universal Time) and a negative number for any time before
`getTimezoneOffset()`		Returns time zone offset in mins for locale
`toDateString()`		Returns "date" as a human-readable string
`toTimeString()`		Returns "time" as a human-readable string

Questions ?