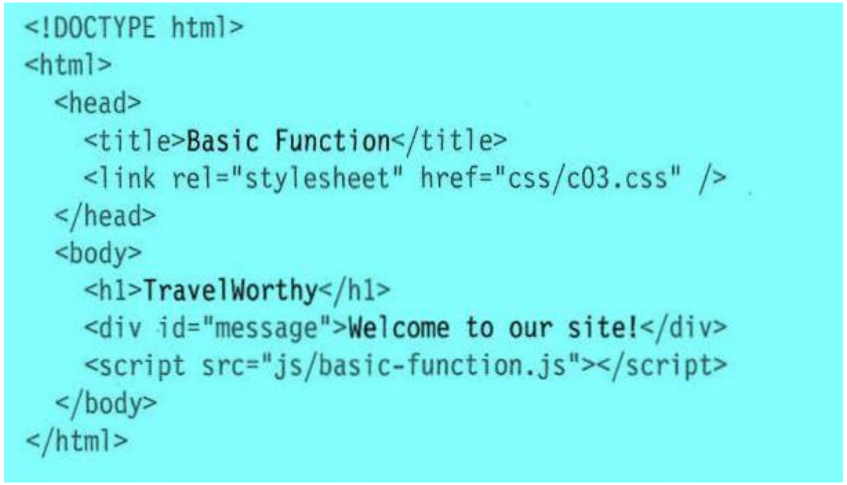
JAVASCRIPT: FUNCTIONS, OBJECTS AND ANIMATION

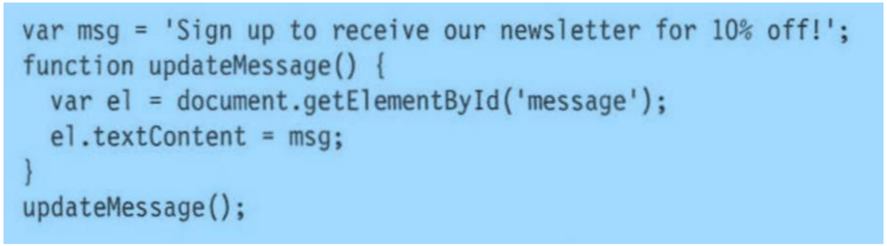
Asst. Lect. Ali Al-khawaja

WHAT IS A FUNCTION?

- 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 set of st atements).



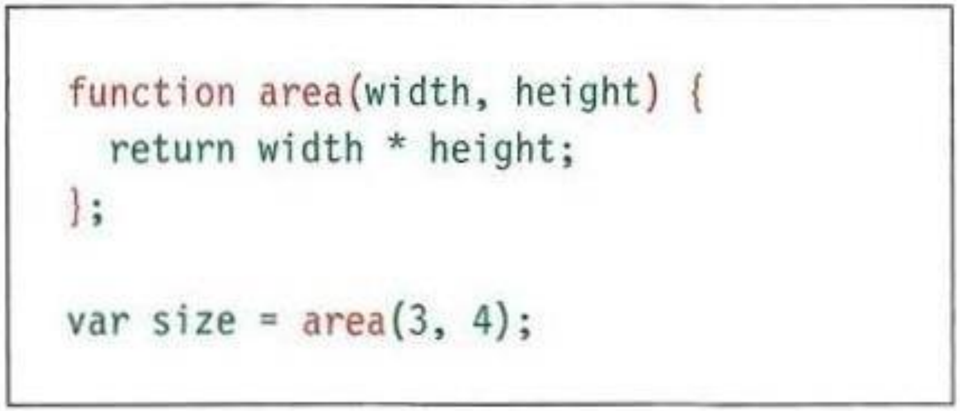
```
<!DOCTYPE html>
<html>
  <head>
    <title>Basic Function</title>
    <link rel="stylesheet" href="css/c03.css" />
  </head>
  <body>
    <h1>TravelWorthy</h1>
    <div id="message">Welcome to our site!</div>
    <script src="js/basic-function.js"></script>
  </body>
</html>
```



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

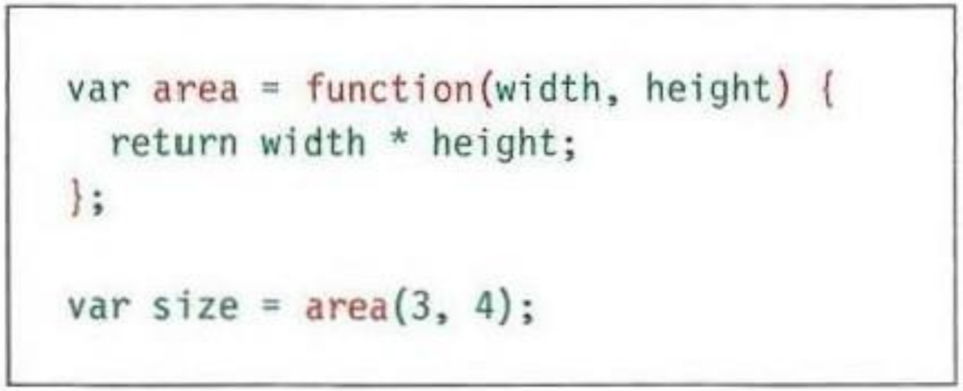
FUNCTION DECLARATION & FUNCTION EXPRESSION

Function Declaration



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

var size = area(3, 4);
```

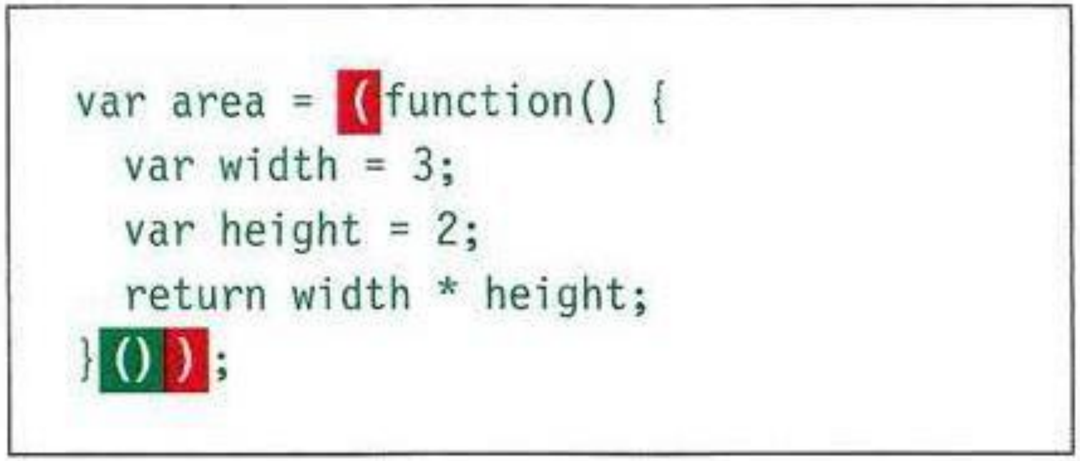
Function Expression



```
var area = function(width, height) {
  return width * height;
};

var size = area(3, 4);
```

IMMEDIATELY INVOKED FUNCTION EXPRESSIONS (LLFE)

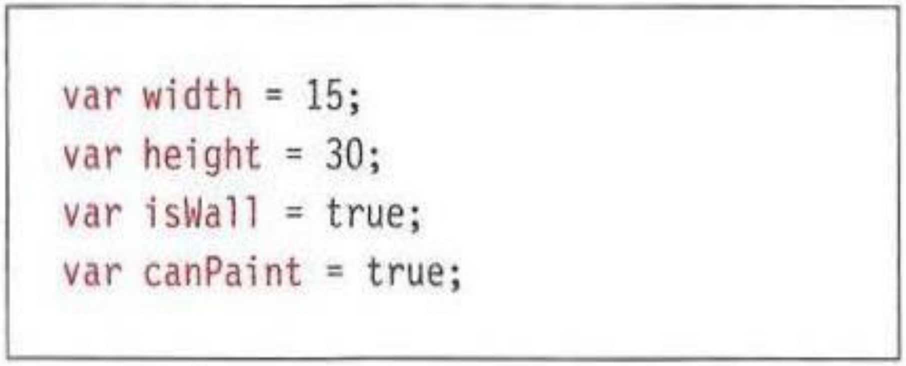
- Pronounced "iffy," these functions are not given a name. Instead, they are executed once as the interpreter comes across them.
- Below, the variable called area will hold the value returned from the function (rather than storing the function itself so that it can be called later).



```
var area = (function() {
  var width = 3;
  var height = 2;
  return width * height;
}());
```

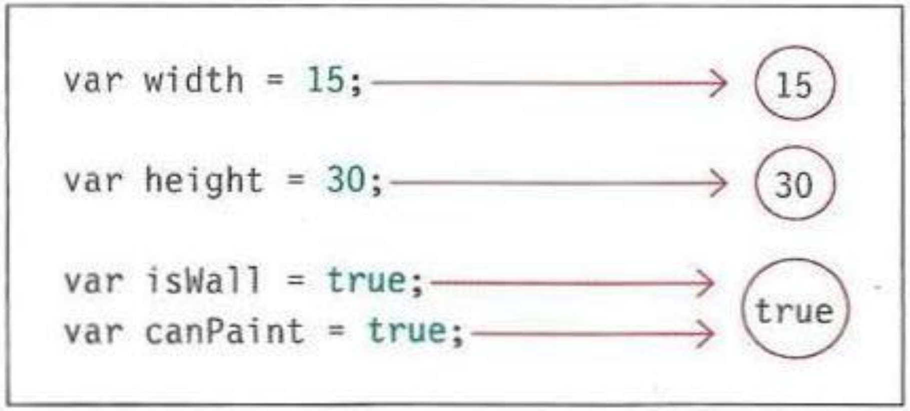
MEMORY & VARIABLES

Each variable that you declare takes up memory. The more variables a browser has to remember, the more memory your script requires to run.
Scripts that require a lot of memory can perform slower, which in turn makes your web page take longer to respond to the user.



```
var width = 15;
var height = 30;
var isWall = true;
var canPaint = true;
```

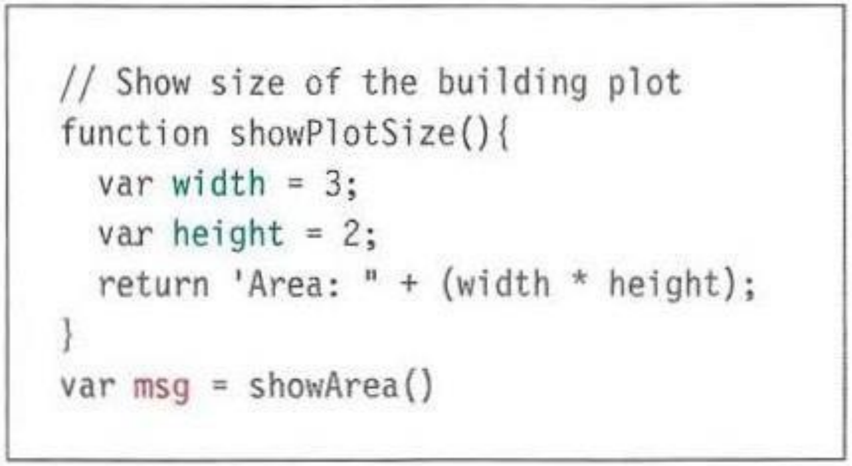
A variable actually references a value that is stored in memory. The same value can be used with more than one variable.



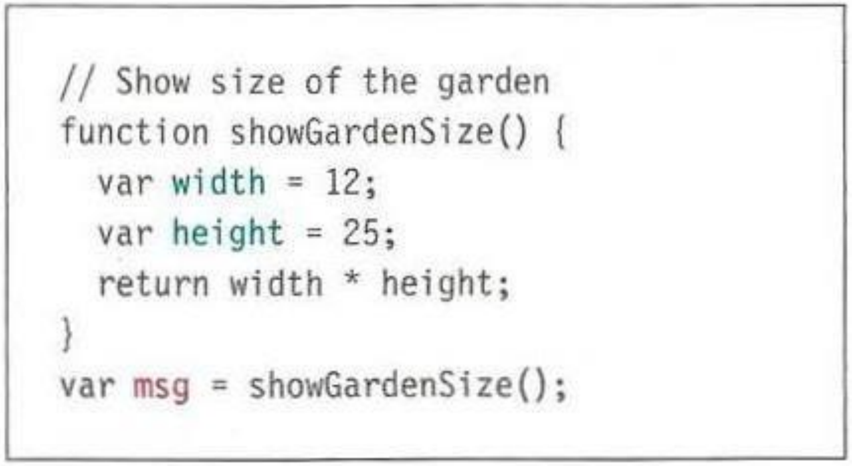
```
var width = 15;
var height = 30;
var isWall = true;
var canPaint = true;
```

NAMING COLLISIONS

You might think you would avoid naming collisions; after all you know which variables you are using. But many sites use scripts written by several people. If an HTML page uses two JavaScript files, and both have a global variable with the same name, it can cause errors. Imagine a page using these two scripts:



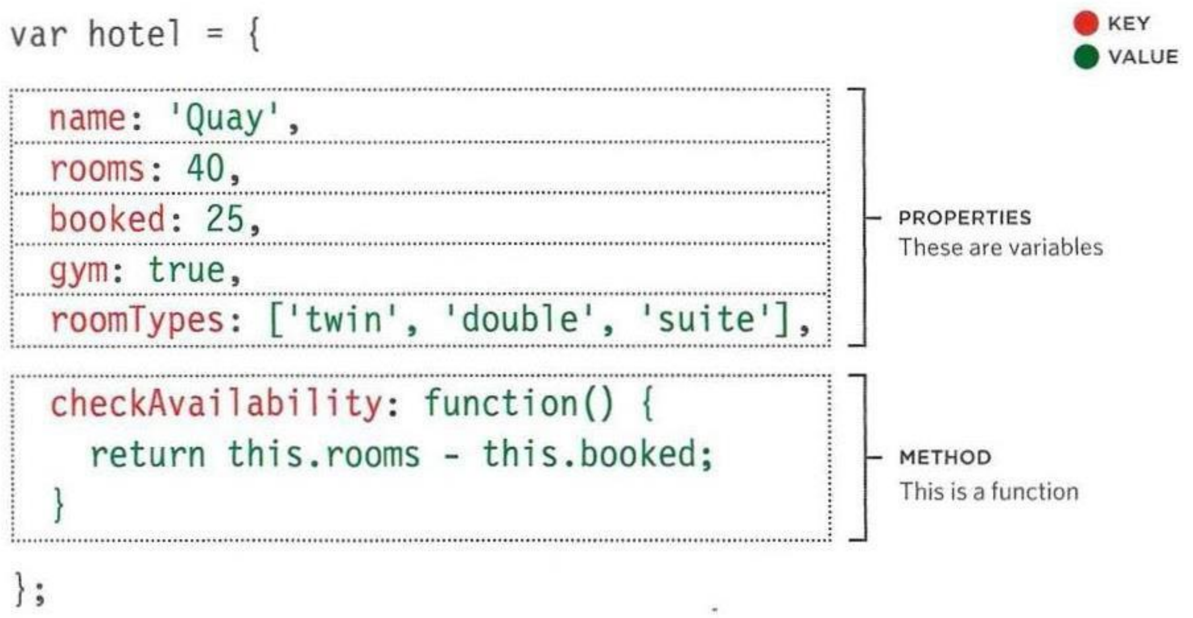
```
// Show size of the building plot
function showPlotSize(){
  var width = 3;
  var height = 2;
  return 'Area: " + (width * height);
}
var msg = showArea()
```



```
// Show size of the garden
function showGardenSize() {
  var width = 12;
  var height = 25;
  return width * height;
}
var msg = showGardenSize();
```

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
    - IN AN OBJECT: FUNCTIONS BECOME KNOWN AS METHODS



```
var hotel = {
  name: 'Quay',
  rooms: 40,
  booked: 25,
  gym: true,
  roomTypes: ['twin', 'double', 'suite'],

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

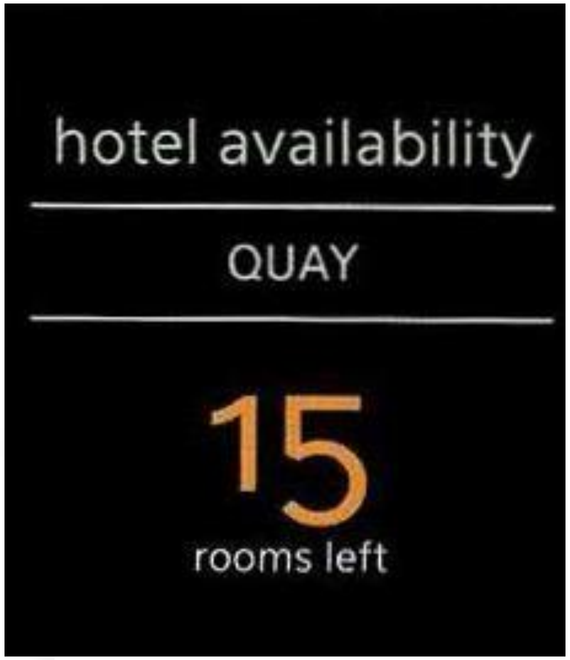
CREATING· OBJECTS USING LITERAL NOTATION

To access a property of this object, the object name is followed by a dot (the period symbol) and the name of the property that you want. Similarly, to use the method, you can use the object name followed by the method name.

hotel.checkAvailability().

If the method needs parameters, you can supply them in the parentheses (just like you can pass arguments to a function).

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

var elName = document.getElementById('hotelName');
elName.textContent = hotel.name;

var elRooms = document.getElementById('rooms');
elRooms.textContent = hotel.checkAvailability();
```



CREATE & ACCESS OBJECTS CONSTRUCTOR NOTATION

```
function Hotel(name, rooms, booked) {
  this.name = name;
  this.rooms = rooms;
  this.booked = booked;
  this.checkAvailability = function() {
    return this.rooms - this.booked;
  };
}

var quayHotel = new Hotel('Quay', 40, 25);
var parkHotel = new Hotel('Park', 120, 77);

var details1 = quayHotel.name + ' rooms: ';
    details1 += quayHotel.checkAvailability();
var elHotel1 = document.getElementById('hotel1');
elHotel1.textContent = details1;

var details2 = parkHotel.name + ' rooms: ';
    details2 += parkHotel.checkAvailability();
var elHotel2 = document.getElementById('hotel2');
elHotel2.textContent = details2;
```

To create multiple objects on the same page, here is an example that shows room availability in two hotels.
First, a constructor function defines a template for the hotels. Next, two different instances of this type of hotel object are created. The first represents a hotel called Quay and the second a hotel called Park.
Having created instances of these objects, you can then access their properties and methods using the same dot notation that you use with all other objects. For each hotel, a variable is created to hold the hotel name, followed by space, and the word rooms. The line after it adds to that variable with the number of available rooms in that hotel.

WHAT ARE BUILT-IN OBJECTS?

BROWSER OBJECT MODEL

The Browser Object Model contains objects that represent the current browser window or tab. It contains objects that model things like browser history and the device's screen.

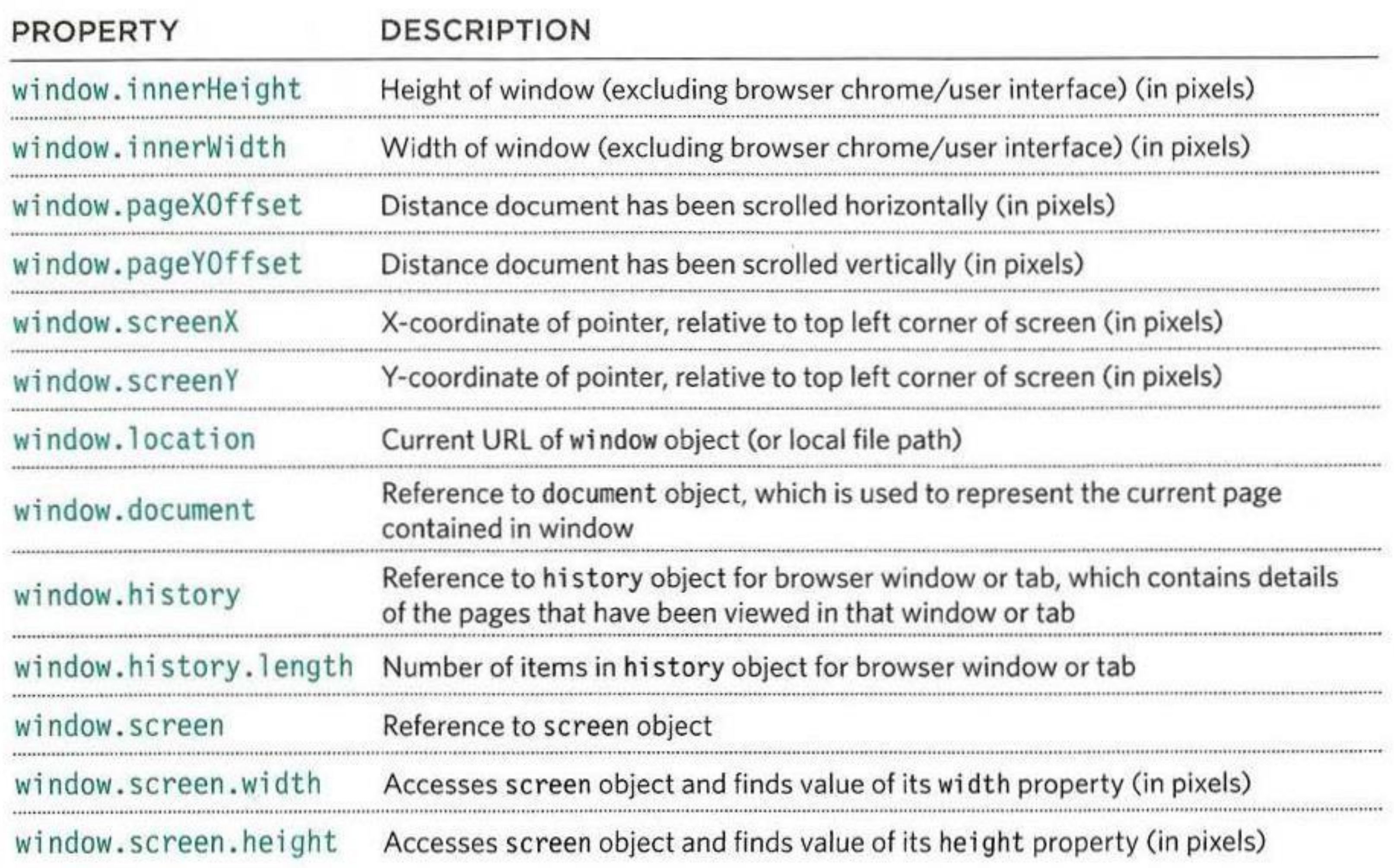
DOCUMENT OBJECT MODEL

The Document Object Model uses objects to create a representation of the current page. It creates a new object for each element (and each individual section of text) within the page.

GLOBAL JAVASCRIPT OBJECTS

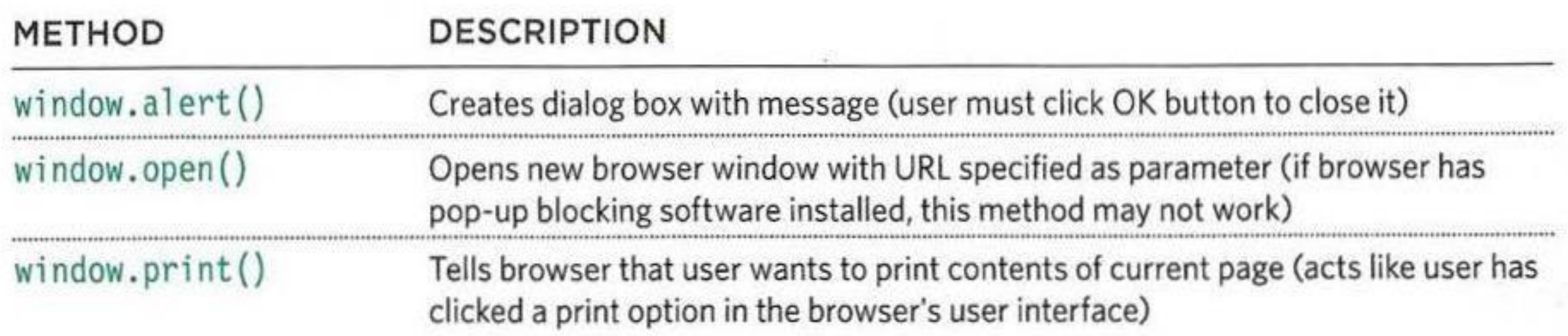
The global JavaScript objects represent things that the JavaScript language needs to create a model of. For example, there is an object that deals only with dates and times.

THE BROWSER OBJECT MODEL:THE WINDOW OBJECT

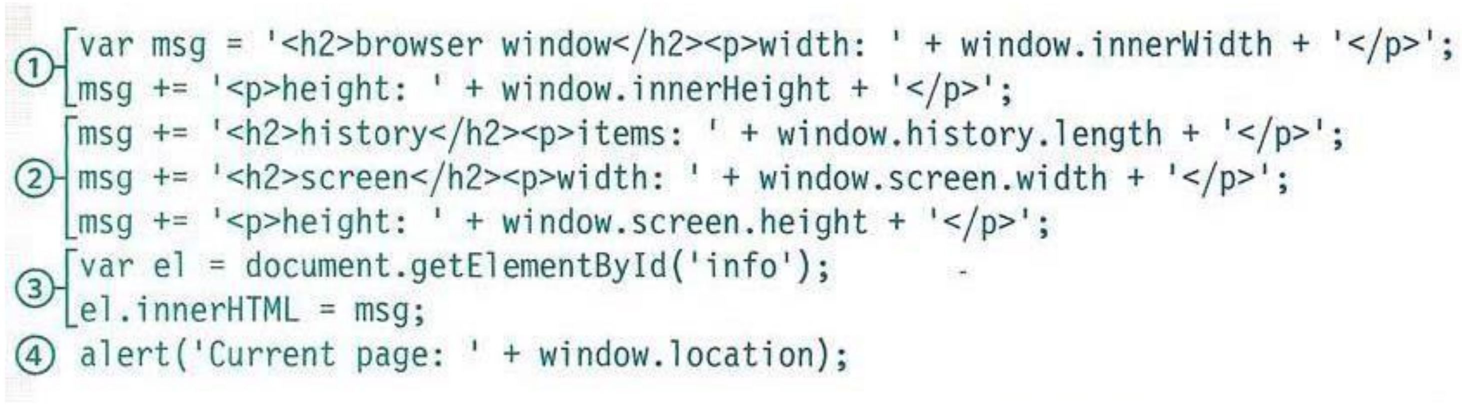


PROPERTY	DESCRIPTION
`window.innerHeight`	Height of window (excluding browser chrome/user interface) (in pixels)
`window.innerWidth`	Width of window (excluding browser chrome/user interface) (in pixels)
`window.pageXOffset`	Distance document has been scrolled horizontally (in pixels)
`window.pageYOffset`	Distance document has been scrolled vertically (in pixels)
`window.screenX`	X-coordinate of pointer, relative to top left corner of screen (in pixels)
`window.screenY`	Y-coordinate of pointer, relative to top left corner of screen (in pixels)
`window.location`	Current URL of `window` object (or local file path)
`window.document`	Reference to `document` object, which is used to represent the current page contained in window
`window.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
`window.history.length`	Number of items in `history` object for browser window or tab
`window.screen`	Reference to `screen` object
`window.screen.width`	Accesses `screen` object and finds value of its `width` property (in pixels)
`window.screen.height`	Accesses `screen` object and finds value of its `height` property (in pixels)

THE BROWSER OBJECT MODEL:THE WINDOW OBJECT



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)



```
① var msg = '<h2>browser window</h2><p>width: ' + window.innerWidth + '</p>';
  msg += '<p>height: ' + window.innerHeight + '</p>';
② msg += '<h2>history</h2><p>items: ' + window.history.length + '</p>';
  msg += '<h2>screen</h2><p>width: ' + window.screen.width + '</p>';
  msg += '<p>height: ' + window.screen.height + '</p>';
③ var el = document.getElementById('info');
  el.innerHTML = msg;
④ alert('Current page: ' + window.location);
```

browser window

width : 1232

height: 649

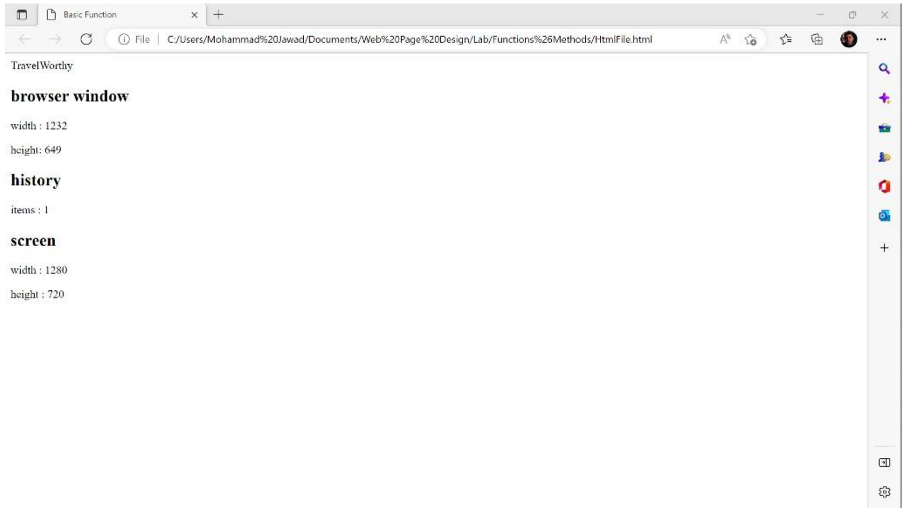
history

items : 1

screen

width : 1280

height : 720



Basic Function

File | C:/Users/Mohammad%20Jawad/Documents/Web%20Page%20Design/Lab/Functions%26Methods/HtmlFile.html

TravelWorthy

browser window

width : 1232

height: 649

history

items : 1

screen

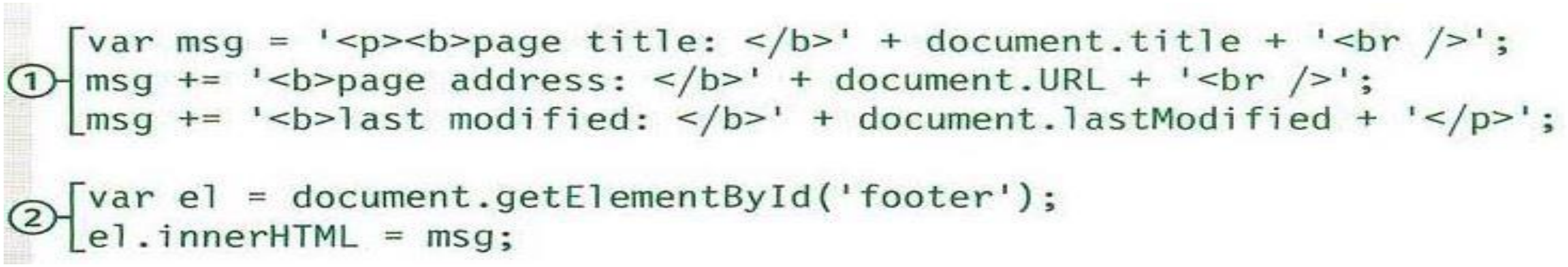
width : 1280

height : 720

THE DOCUMENT OBJECT MODEL

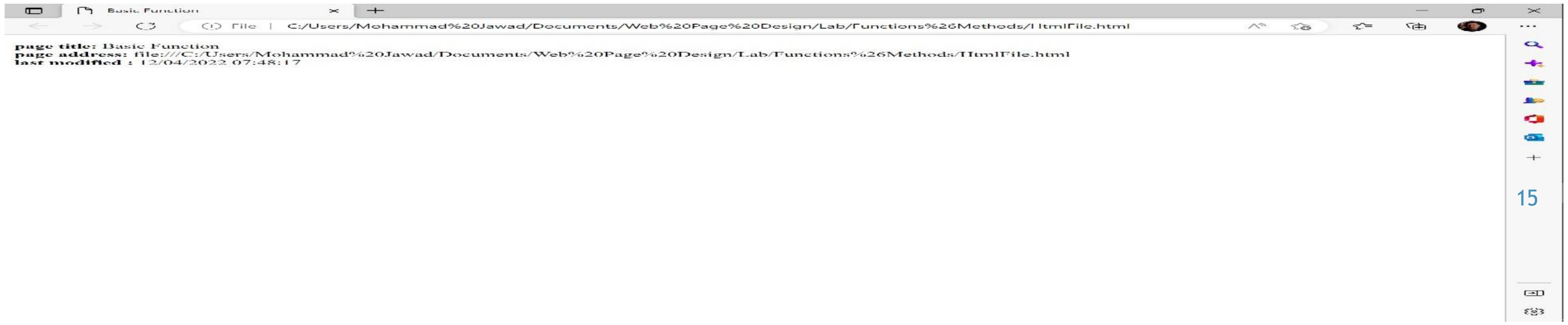
PROPERTY	DESCRIPTION
`document.title`	Title of current document
`document.lastModified`	Date on which document was last modified
`document.URL`	Returns string containing URL of current document
`document.domain`	Returns domain of current document

METHOD	DESCRIPTION
`document.write()`	Writes text to document
`document.getElementById()`	Returns element, if there is an element with the value of the `id` attribute that matches
`document.querySelectorAll()`	Returns list of elements that match a CSS selector, which is specified as a parameter
`document.createElement()`	Creates new element
`document.createTextNode()`	Creates new text node



```
var msg = '<p><b>page title: </b>' + document.title + '<br />';
msg += '<b>page address: </b>' + document.URL + '<br />';
msg += '<b>last modified: </b>' + document.lastModified + '</p>';

var el = document.getElementById('footer');
el.innerHTML = msg;
```

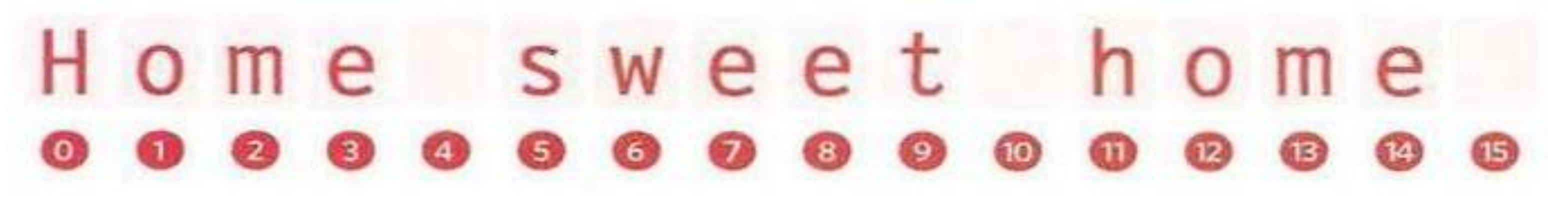
page title: Basic Function
page address:
file:///C:/Users/Mohammad%20Jawad/Documents/Web%20Page%20Design/Lab/Functions%26Methods/HtmlFile.html
last modified : 12/04/2022 07:48:17

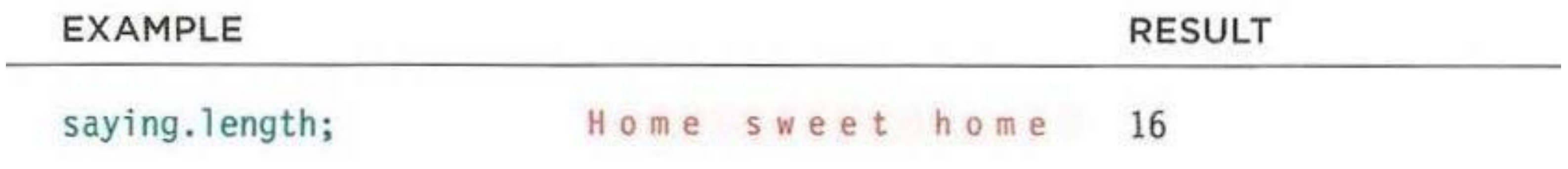


Basic Function
File | C:/Users/Mohammad%20Jawad/Documents/Web%20Page%20Design/Lab/Functions%26Methods/HtmlFile.html

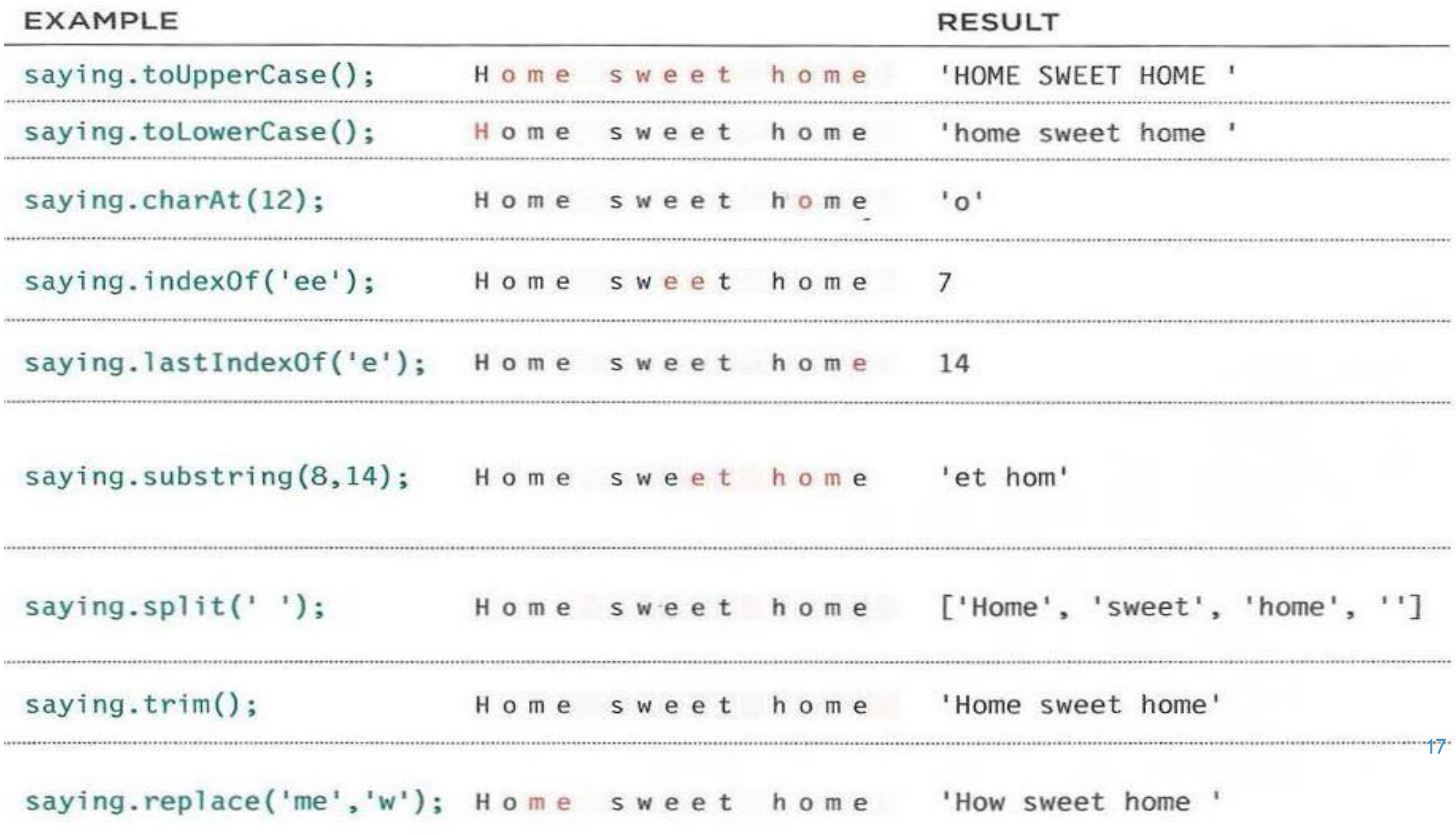
page title: Basic Function
page address: file:///C:/Users/Mohammad%20Jawad/Documents/Web%20Page%20Design/Lab/Functions%26Methods/HtmlFile.html
last modified : 12/04/2022 07:48:17

GLOBAL OBJECTS: STRING OBJECT



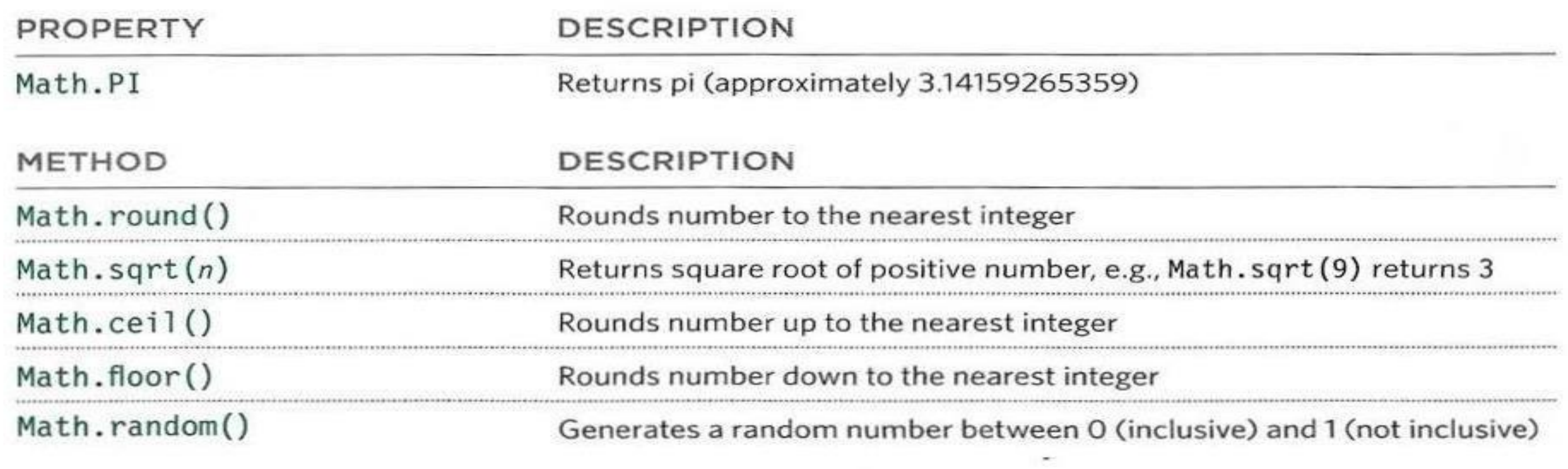


EXAMPLE		RESULT
`saying.length;`	Home sweet home	16



EXAMPLE		RESULT
`saying.toUpperCase();`	H o m e s w e e t h o m e	'HOME SWEET HOME '
`saying.toLowerCase();`	H o m e s w e e t h o m e	'home sweet home '
`saying.charAt(12);`	H o m e s w e e t h o m e	'o'
`saying.indexOf('ee');`	H o m e s w e e t h o m e	7
`saying.lastIndexOf('e');`	H o m e s w e e t h o m e	14
`saying.substring(8,14);`	H o m e s w e e t h o m e	'et hom'
`saying.split(' ');`	H o m e s w e e t h o m e	['Home', 'sweet', 'home', '']
`saying.trim();`	H o m e s w e e t h o m e	'Home sweet home'
`saying.replace('me','w');`	H o m e s w e e t h o m e	'How sweet home '

GLOBAL OBJECTS: MATH OBJECT

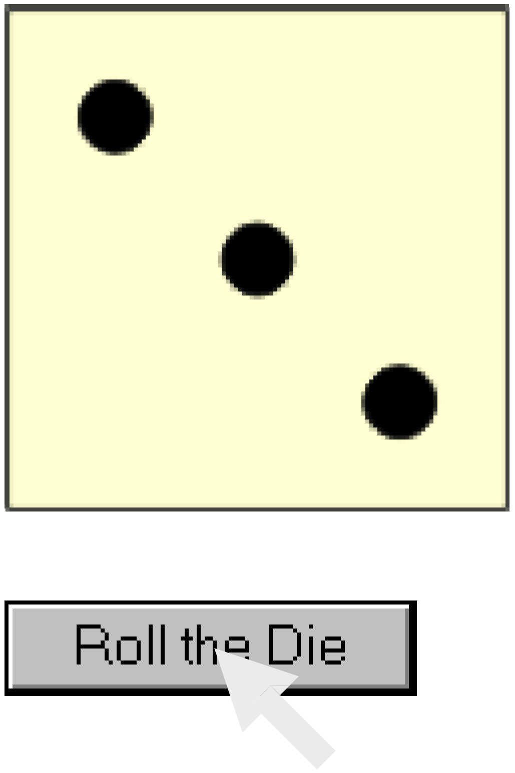


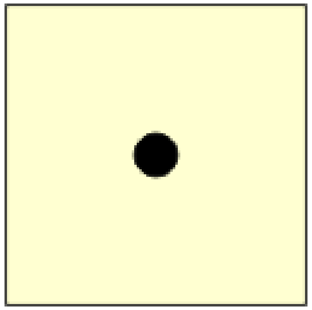
PROPERTY	DESCRIPTION
`Math.PI`	Returns pi (approximately 3.14159265359)

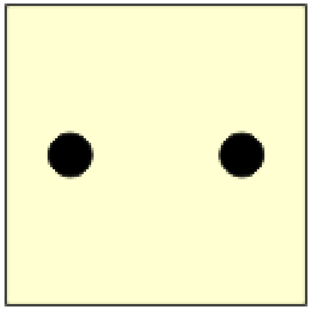
METHOD	DESCRIPTION
`Math.round()`	Rounds number to the nearest integer
`Math.sqrt(n)`	Returns square root of positive number, e.g., `Math.sqrt(9)` returns 3
`Math.ceil()`	Rounds number up to the nearest integer
`Math.floor()`	Rounds number down to the nearest integer
`Math.random()`	Generates a random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

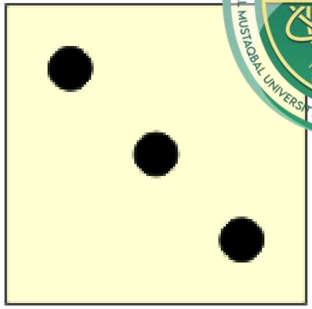
EXAMPLES

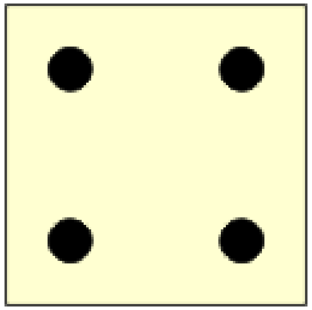
- Let Us now do some examples for more understanding, and the first example will be (Role the dice Example)

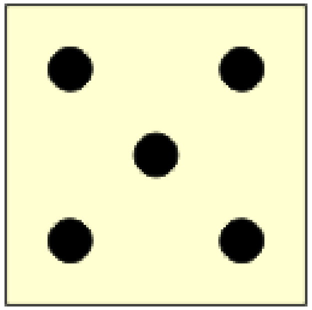

Roll the Die

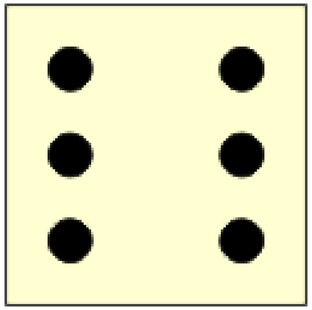

die1.gif


die2.gif


die3.gif


die4.gif


die5.gif


die6.gif

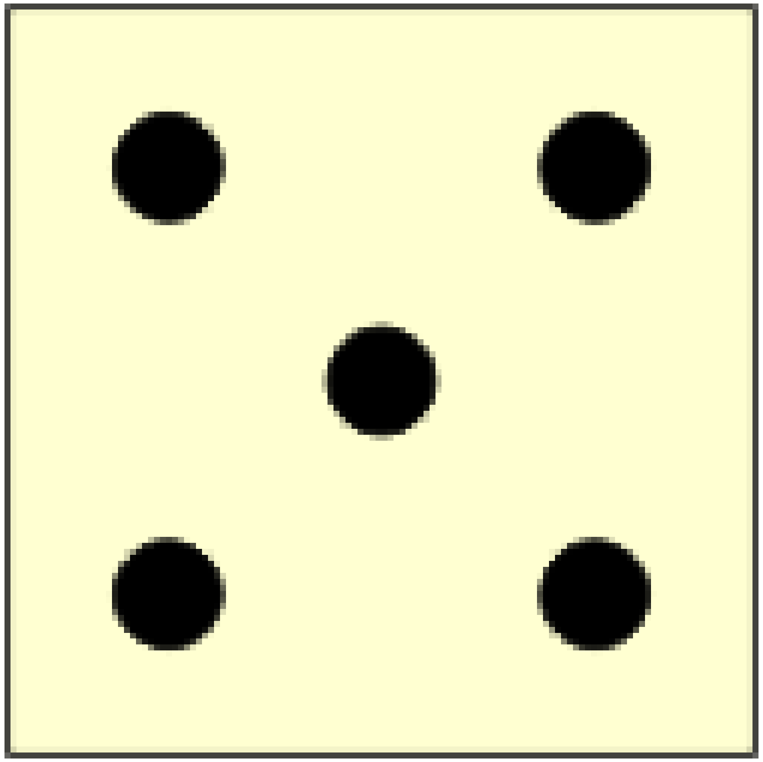
```
<HTML>
<HEAD>
        <TITLE>Roll the Die</TITLE>
        <SCRIPT>
            JavaScript Code
        </SCRIPT>
</HEAD>

<BODY >
        HTML Code
</BODY>
</HTML>
```

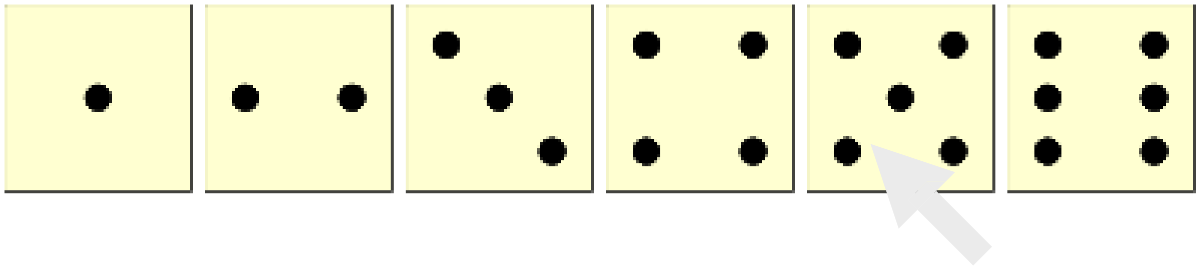
```
<IMG name="die" src="die6.gif">

<FORM>
        <INPUT type="button" value="Roll the Die"
            onClick="rollDie( )">
</FORM>
```

```
dieImg = new Array( 7 ) ;
for( k = 1; k < 7; k = k + 1 ) { //Preload images
        dieImg[ k ] = new Image( ) ;
        dieImg[ k ].src = "die" + k + ".gif" ;
}
function rollDie( ) {
        dieN = Math.ceil( 6 * Math.random( ) ) ;
        document.die.src = dieImg[ dieN ].src ;
}
```

ANOTHER EXAMPLE

- Develop a Web page that displays six thumbnail images and a main image
- The main image should change to a larger version of the thumbnail as soon as the mouse moves over on a thumbnail image





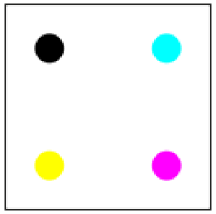
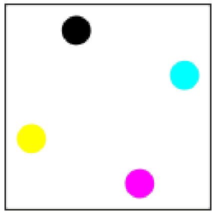
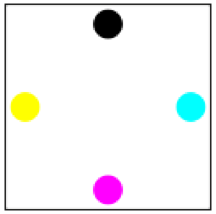
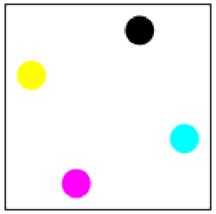
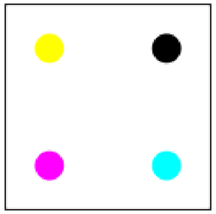
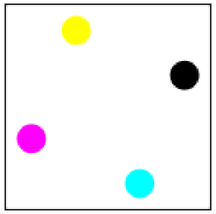
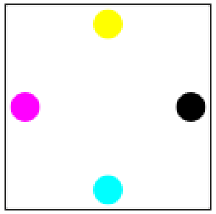
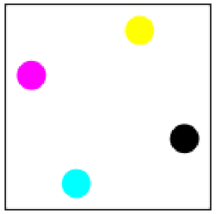
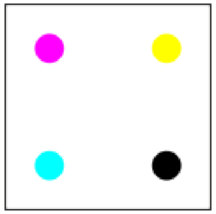
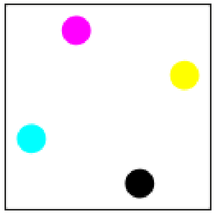
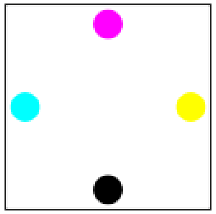
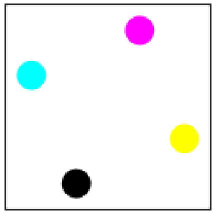
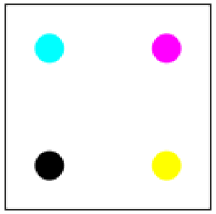
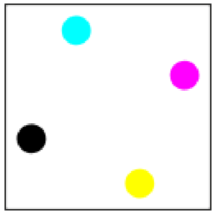
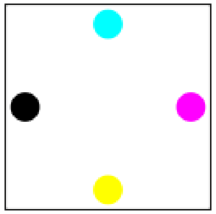
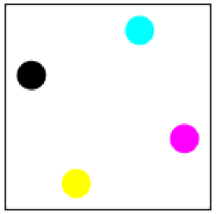
```
<HTML>
<HEAD>
        <TITLE>Image Selector</TITLE>
        <SCRIPT>
            JavaScript Code
        </SCRIPT>
</HEAD>

<BODY >
        HTML Code
</BODY>
</HTML>
```

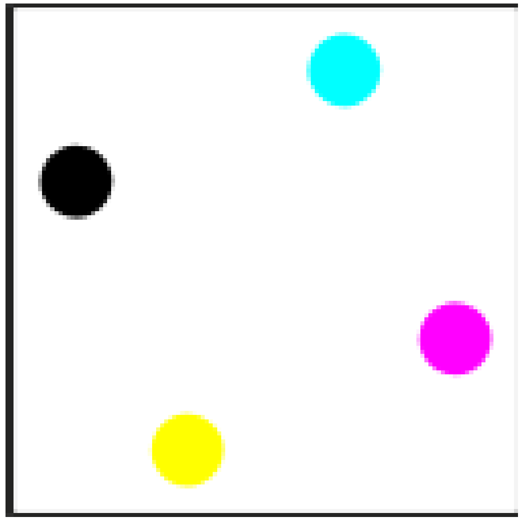
```
dieImg = new Array( 7 ) ;

for( k = 1; k < 7; k = k + 1 ) { // Preload images

        dieImg[ k ] = new Image( ) ;

        dieImg[ k ].src = "die" + k + ".gif" ;

}
```

```
<IMG name="big" src="die6.gif" width="252"
    height="252"><P>


<IMG src="die1.gif" width="63" height="63"

    onMouseOver= "document.big.src=dieImg[ 1 ].src">

    …

    …

<IMG src="die6.gif" width="63" height="63"

    onMouseOver= document.big.src=dieImg[ 6 ].src">
```

ANIMATION EXAMPLE 2

- Take 16 images and cycle through them to create an animation effect

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16




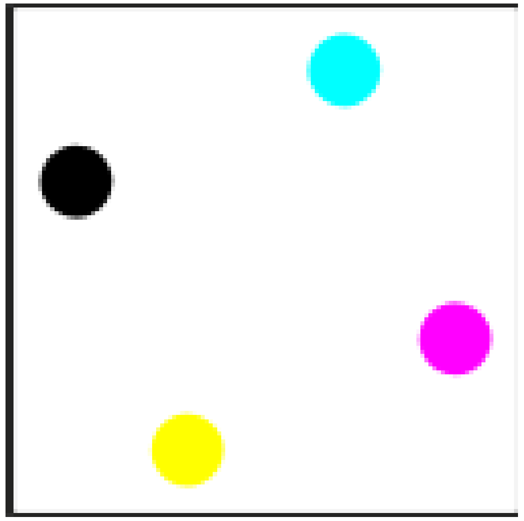
```
<HTML>
<HEAD>
        <TITLE>Animation 1</TITLE>
        <SCRIPT>
            JavaScript Code
        </SCRIPT>
</HEAD>
<BODY >
        HTML Code
</BODY>
</HTML>
```

```
<CENTER>

   <IMG name="circle" src="circle1.gif"
      onLoad="setTimeout( 'circulate( )', gap )">

</CENTER>
```

setTimeout( ) executes *circulate( )* once after a delay of *gap* milliseconds

```
gap = 100 ;
imageN = 1 ;
circImg = new Array( 17 ) ;

for( k = 1; k < 17; k = k + 1 ) { // Preload images

        circImg[ k ] = new Image( ) ;

        circImg[ k ].src = "circle" + k + ".gif" ;
}
```

```
function circulate( ) {

        document.circle.src = circImg[ imageN ].src ;

        imageN = imageN + 1 ;

        if( imageN > 16 )
            imageN = 1 ;
}
```



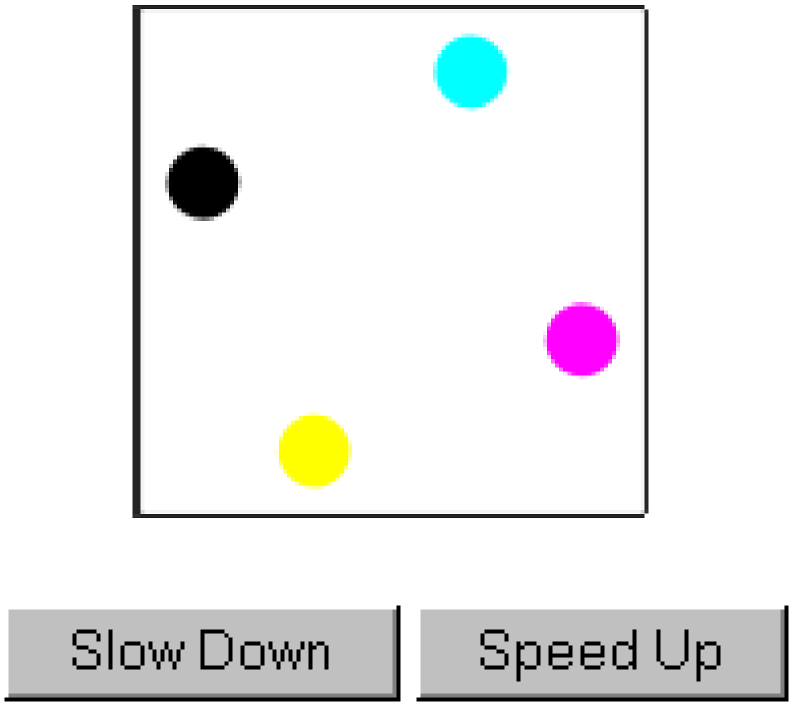
ANIMATED GIFS

- We could have saved the 16 gif images of the previous example in a single file in the form of an animated gif, and then used it in a regular <IMG> tag to display a moving image

- However, JavaScript provides better control over the sequencing and the gap between the individual images

ANIMATION EXAMPLE 2

- Take 16 images and cycle through them to create an animation effect
- Provide buttons to slow down or speed up the animation


Slow Down
Speed Up

```
<HTML>
<HEAD>
        <TITLE>Animation 2</TITLE>
        <SCRIPT>
            JavaScript Code
        </SCRIPT>
</HEAD>

<BODY >
        HTML Code
</BODY>
</HTML>
```

```
<CENTER>
    <IMG name="circle" src="circle1.gif"    onLoad="setTimeout( 'circulate( )', gap
    )">
</CENTER>

<FORM>
    <INPUT type="button" value="Slow Down"
        onClick="slowDown( )">
    <INPUT type="button" value="Speed Up"
        onClick="speedUp( )">
</FORM>
```

```
gap = 100 ;
imageN = 1 ;
circImg = new Array( 17 ) ;

for( k = 1; k < 17; k = k + 1 ) { // Preload images

        circImg[ k ] = new Image( ) ;

        circImg[ k ].src = "circle" + k + ".gif" ;
}
```

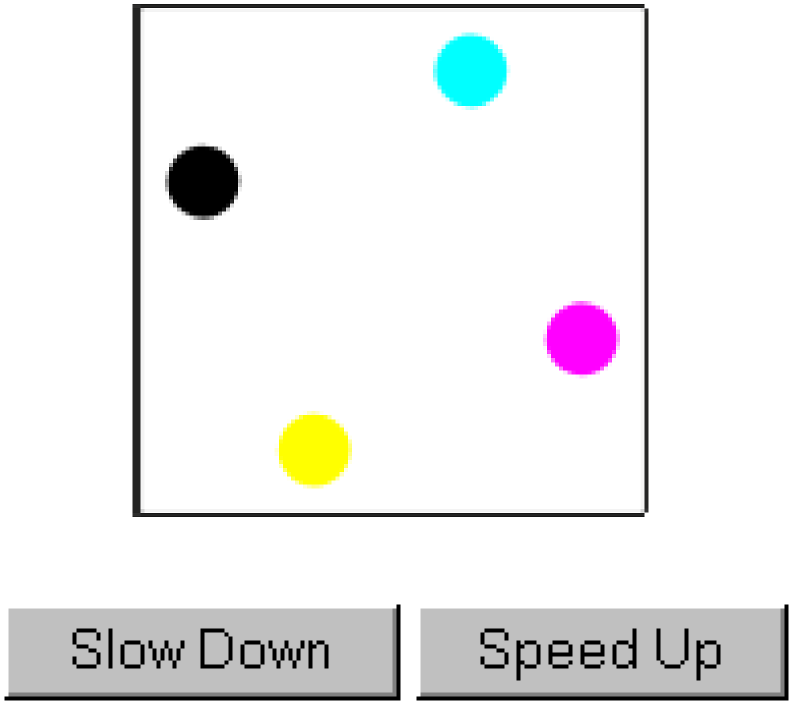
No
change

```
function circulate( ) {

        document.circle.src =
            circImg[ imageN ].src ;

        imageN = imageN + 1 ;

        if( imageN > 16 )
            imageN = 1 ;
}
```

No change

```
function slowDown( ) {
        gap = gap + 20 ;
        if( gap > 4000 )
                gap = 4000 ;
}
function speedUp( ) {
        gap = gap - 20 ;
        if( gap < 0 )
                gap = 0 ;
}
```

Two new functions


Slow Down
Speed Up