

CSCI 1720  
JavaScript – Part 2

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

1

---

---

---

---

---

---

---

---

JavaScript Operators

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

2

---

---

---

---

---

---

---

---

JS Operators

JavaScript operators are identical to Java's

- + Addition
- Subtraction
- \* Multiplication
- / Division
- % Modulus
- ++ Increment
- Decrement

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

3

---

---

---

---

---

---

---

---

### JS Assignment Operators

JavaScript operators are identical to Java's

Operator	Example	Same as
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

4

---

---

---

---

---

---

---

---

---

---

### JS String Operators

The += assignment operator can also be used to add (concatenate) strings

```

<p>The assignment operator += can concatenate strings.</p>
<p id="demo"></p>
<script>
txt1 = "One ring to rule them all, ";
txt1 += "One ring to find them";
document.getElementById("demo").innerHTML = txt1;
</script>

```



The assignment operator += can concatenate strings.  
One ring to rule them all, One ring to find them

5

---

---

---

---

---

---

---

---

---

---

### JS Adding Strings and Numbers

Adding two numbers, will return the sum, but adding a number and a string will return a string

```

<p>Adding a number and a string, returns a string.</p>
<p id="demo"></p>
<script>
var x = 5 + 5;
var y = "5" + 5;
var z = "hello" + 5;
document.getElementById("demo").innerHTML =
x + "demo" + y + "demo" + z;
</script>

```



Adding a number and a string, returns a string.  
10  
55  
Hello5

6

---

---

---

---

---

---

---

---

---

---

### JS Comparison Operators

JavaScript operators are identical to Java's, with a couple of additional ones

Operator	Description
==	equal to
===	equal value & equal type
!=	not equal
!==	not equal value or type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

7

---

---

---

---

---

---

---

---

---

---

### JS Logical Operators

JavaScript operators are identical to Java's, with a couple of additional ones

Operator	Description
&&	logical and
	logical or
!	Logical not

8

---

---

---

---

---

---

---

---

---

---

### JavaScript Data Types

9

---

---

---

---

---

---

---

---

---

---

JS Data Types

JavaScript variables can hold many data types: numbers, strings, objects and more

```
var length = 16;           // Number
var lastname = "Johnson"; // String
var x = {firstname:"John", lastname:"Doe"}; // Object
```

In programming, data types is an important concept  
To be able to operate on variables, it is important to know something about the type

Without data types, a computer cannot safely solve this

```
var x = 16 + "Volvo";
```

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

10

---

---

---

---

---

---

---

---

---

---

JS Data Types

JavaScript has dynamic types

This means that the same variable can be used to hold different data types

```
var x;           // Now x is undefined
var x = 5;       // Now x is a Number
var x = "John";  // Now x is a String
```

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

11

---

---

---

---

---

---

---

---

---

---

JS Data Types

JavaScript has only one type of numbers

Numbers can be written with, or without decimals

```
var x1 = 34.00; // Written with decimals
var x2 = 34;    // Written without decimals
```

Extra large or extra small numbers can be written with scientific (exponential) notation

```
var y = 123e5; // 12300000
var z = 123e-5; // 0.00123
```

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

12

---

---

---

---

---

---

---

---

---

---

JS Data Types

Booleans can only have two values: true or false

```
var x = true;
var y = false;
```

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

13

---

---

---

---

---

---

---

---

JS Data Types

JavaScript arrays are written with square brackets  
Array items are separated by commas  
The following code declares (creates) an array called cars, containing three items (car names)

```
var cars = ["Saab", "Volvo", "BMW"];
```

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

14

---

---

---

---

---

---


---

---

JS Data Types

JavaScript objects are written with braces  
Object properties are written as name:value pairs, separated by commas

```
<script>
var person = {
  firstName : "John",
  lastName  : "Doe",
  age       : 50,
  eyeColor  : "blue"
};
document.getElementById("demo").innerHTML =
  person.firstName + " is " + person.age +
  " years old.";
</script>
```

 John is 50 years old.

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

15

---

---

---

---

---

---

---

---

### JS Data Types – typeof Operator

You can use the JavaScript `typeof` operator to find the type of a JavaScript variable

The `typeof` operator returns the type of a variable or an expression

```

typeof "" // Returns "string"
typeof "John" // Returns "string"
typeof "John Doe" // Returns "string"
typeof 0 // Returns "number"
typeof 314 // Returns "number"
typeof 3.14 // Returns "number"
typeof (3) // Returns "number"
typeof (3 + 4) // Returns "number"
    
```

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

16

---

---

---

---

---

---

---

---

---

---

### JS Data Types – Primitive Data

A primitive data value is a single simple data value with no additional properties and methods

The `typeof` operator can return one of these primitive types:

```

string      typeof "John" // Returns "string"
number     typeof 3.14 // Returns "number"
Boolean    typeof true // Returns "boolean"
null       typeof false // Returns "boolean"
undefined
    
```

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

17

---

---

---

---

---

---

---

---

---

---

### JS Data Types

The `typeof` operator can return one of two complex types:

```

function
object
typeof [1,2,3,4] // Returns "object" (not "array", see note below)
typeof {name:'John', age:34} // Returns "object"
typeof function myfunc(){} // Returns "function"
    
```

The `typeof` operator returns **object** for arrays because in JavaScript arrays are objects

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

18

---

---

---

---

---

---

---

---

---

---

JS Data Types

In JavaScript, a variable without a value, has the value undefined  
The `typeof` is also undefined

```
var person;           // Value is undefined, type is undefined
```

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

19

---

---

---

---

---

---

---

---

JS Data Types

In JavaScript, a variable without a value, has the value undefined  
The `typeof` is also undefined

```
var person;           // Value is undefined, type is undefined
```

An empty value has nothing to do with undefined  
An empty string variable has both a value and a type

```
var car = "";         // The value is "", the typeof is "string"
```

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

20

---

---

---

---

---

---

---

---

JS Data Types

In JavaScript null is "nothing"  
It is supposed to be something that doesn't exist  
Unfortunately, in JavaScript, the data type of null is an object  
You can consider it a bug in JavaScript that `typeof null` is an object.  
It should be null

You can empty an object by setting it to null

```
var person = null;    // Value is null, but type is still an object
```

You can also empty an object by setting it to undefined

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

21

---

---

---

---

---

---

---

---

## JS Data Types

### Difference Between Undefined and Null

```
typeof undefined // undefined
typeof null      // object
null === undefined // false
null == undefined // true
```

22

---

---

---

---

---

---

---

---

## JavaScript Functions

23

---

---

---

---

---

---

---

---

## JS Functions

A JavaScript function is a block of code designed to perform a particular task

A JavaScript function is executed when **something** invokes it (calls it)

```
function myFunction(p1, p2) {
  return p1 * p2; // The function returns the product of p1 and p2
}
```

24

---

---

---

---

---

---

---

---



### JS Function Syntax

A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ()

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables)

The parentheses may include parameter names separated by commas (parameter1, parameter2, ...)

The code to be executed, by the function, is placed inside brackets: {}

```
function name(parameter1, parameter2, parameter3) {  
  code to be executed  
}
```

25

---

---

---

---

---

---

---

---

---

---

### JS Function Syntax

Function parameters are the names listed in the function definition

Function arguments are the real values received by the function when it is invoked

Inside the function, the arguments (the parameters) behave as local variables

A Function is much the same as a Procedure or a Subroutine, in other programming languages

26

---

---

---

---

---

---

---

---

---

---

### JS Function Invocation

The code inside the function will execute when "something" invokes (calls) the function:

- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self invoked)

27

---

---

---

---

---

---

---

---

---

---

### JS Function Return

JavaScript reaches a return statement, the function will stop executing

If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement

Functions often compute a return value. The return value is returned back to the caller

28

---

---

---

---

---

---

---

---

---

---

### JS Function Example

```

<p>
  This example calls a function<br>
  which performs a calculation<br>
  and returns the result.
</p>
<p id="demo"></p>
<script>
function myFunction(a, b) {
  return a * b;
}
document.getElementById("demo").innerHTML = myFunction(4, 3);
</script>

```



This example calls a function which performs a calculation and returns the result: 12

29

---

---

---

---

---

---

---

---

---

---

### JS – Why Functions?

You can reuse code: Define the code once, and use it many times

You can use the same code many times with different arguments, to produce different results

Functions can be used the same way as you use variables, in all types of formulas, assignments, and calculations

```

var x = toCelsius(77);
var text = "The temperature is " + x + " Celsius";

```

30

---

---

---

---

---

---

---

---

---

---

# JavaScript Scope

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

31

---

---

---

---

---

---

---

---

# JS Scope

Scope is the set of variables you have access to  
 In JavaScript, objects and functions are also variables  
 In JavaScript, scope is the set of variables, objects, and functions you have access to  
 JavaScript has function scope: The scope changes inside functions

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

32

---

---

---

---

---

---

---

---

# JS Scope

Variables declared within a JavaScript function, are **LOCAL** to the function  
 Local variables have **local scope**: They can only be accessed within the function

```

<p>
  The local variable carName cannot be
  accessed from code outside the<br>
  function:
</p>
<p id="demo"></p>
<script>
  myFunction();
  document.getElementById("demo").innerHTML =
  "The type of carName is " + typeof carName;
  function myFunction() {
    var carName = "Volvo";
  }
</script>

```



The local variable carName cannot be accessed from code outside the function:  
 The type of carName is undefined

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

33

---

---

---

---

---

---

---

---

JS Scope

A variable declared outside a function, becomes GLOBAL

A global variable has global scope: All scripts and functions on a web page can access it

```

<p>
  A GLOBAL variable can be  

  accessed from any script or  

  function.
</p>
<p id="demo"></p>
<script>
  var carName = "Volvo";
  myFunction();

  function myFunction() {
    document.getElementById("demo").innerHTML =
      "I can display " + carName;
  }
</script>

```

A GLOBAL variable can be accessed from any script or function.

I can display Volvo

East Tennessee State University Department of Computing CSCI 1720 Intermediate Web Design

34

---

---

---

---

---

---

---

---

---

---

JS Scope

Do **NOT** create global variables unless you intend to

Global variables, in any programming language, are generally a bad idea

East Tennessee State University Department of Computing CSCI 1720 Intermediate Web Design

35

---

---

---

---

---

---

---

---

---

---

JS Scope

With JavaScript, the global scope is the complete JavaScript environment

In HTML, the global scope is the window object; all global variables belong to it

```

<p>
  In HTML, all global variables  

  will become window variables.
</p>
<p id="demo"></p>
<script>
  var carName = "Volvo";

  // code here can use window.carName
  document.getElementById("demo").innerHTML =
    "I can display " + window.carName;
</script>

```

In HTML, all global variables will become window variables.

I can display Volvo

East Tennessee State University Department of Computing CSCI 1720 Intermediate Web Design

36

---

---

---

---

---

---

---

---

---

---

JS Scope

Your global variables (or functions) can overwrite window variables (or functions)

Any function, including the window object, can overwrite your global variables and functions

The lifetime of a JavaScript variable starts when it is declared

Local variables are deleted when the function is completed

In a web browser, global variables are deleted when you close the browser window (or tab), but remains available to new pages loaded into the same window

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

37

---

---

---

---

---

---

---

---

---

---

JavaScript Events

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

38

---

---

---

---

---

---

---

---

---

---

JS Events

HTML events are things that happen to HTML elements

When JavaScript is used in HTML pages, JavaScript can react on these events

An HTML event can be something the browser does, or something a user does

Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

39

---

---

---

---

---

---

---

---

---

---

JS Events

Often, when events happen, you want to do something  
 JavaScript lets you execute code when events are detected  
 HTML allows event handler attributes, with JavaScript code, to be added to HTML elements

With single quotes:

```
<element event='some JavaScript'>
```

With double quotes:

```
<element event="some JavaScript">
```

East Tennessee State University  
 Department of Computing

CSCI 1720  
 Intermediate Web Design

40

---

---

---

---

---

---

---

---

---

---

JS Events

```
<button onclick="document.getElementById('demo').innerHTML+=Date()">The time is</button>
```

```
<p id="demo"></p>
```

East Tennessee State University  
 Department of Computing

CSCI 1720  
 Intermediate Web Design

41

---

---

---

---

---

---

---

---

---

---

JS Events

Code changes the content of its own element (using `this.innerHTML`)

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

East Tennessee State University  
 Department of Computing

CSCI 1720  
 Intermediate Web Design

42

---

---

---

---

---

---

---

---

---

---

JS Events

JavaScript code is often several lines long  
It is more common to see event attributes calling functions

```

<p>
  Click the button to display the date.
</p>
<button onclick="displayDate()">
  The time is?
</button>
<script>
function displayDate() {
  document.getElementById("demo").
  innerHTML = Date();
}
</script>

```

Click the button to display the date.

Click the button to display the date.

Click the button to display the date.

Mon Aug 07 2017 10:40:22 GMT-0400 (Eastern Daylight Time)

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

43

---

---

---

---

---

---

---

---

---

---

JS Events

Here is a list of some common HTML events

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

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

44

---

---

---

---

---

---

---

---

---

---

JS – What Can JavaScript Do?

Event handlers can be used to handle, and verify, user input, user actions, and browser actions:

- Things that should be done every time a page loads
- Things that should be done when the page is closed
- Action that should be performed when a user clicks a button
- Content that should be verified when a user inputs data

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

45

---

---

---

---

---

---

---

---

---

---

JS – What Can JavaScript Do?

Many different methods can be used to let JavaScript work with events:

- HTML event attributes can execute JavaScript code directly
- HTML event attributes can call JavaScript functions
- You can assign your own event handler functions to HTML elements
- You can prevent events from being sent or being handled

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

46

---

---

---

---

---

---

---

---

JavaScript Conditions

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

47

---

---

---

---

---

---

---

---

JS Conditions

Very often when you write code, you want to perform different actions for different decisions

You can use **conditional statements** in your code to do this

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

48

---

---

---

---

---

---

---

---



### JS Conditions

In JavaScript we have the following conditional statements:

Use **if** to specify a block of code to be executed, if a specified condition is true

Use **else** to specify a block of code to be executed, if the same condition is false

Use **else if** to specify a new condition to test, if the first condition is false

Use **switch** to specify many alternative blocks of code to be executed

49

---

---

---

---

---

---

---

---

---

---

### JS Conditions -- if

Use the if statement to specify a block of JavaScript code to be executed if a condition is true

Note that if is in lowercase letters. Uppercase letters (If or IF) will generate a JavaScript error

```
<script>
if (condition) {
  // code to execute if true;
}
</script>
```

50

---

---

---

---

---

---

---

---

---

---

### JS Conditions -- if

```
<p>Display "Good day!" if the hour is less than 18:00.</p>
<p id="demo">Good Evening!</p>
<script>
if (new Date().getHours() < 18) {
  document.getElementById("demo").innerHTML =
  "Good day!";
}
</script>
```



Display "Good day!" if the hour is less than 18:00:  
Good day!

51

---

---

---

---

---

---

---

---

---

---

### JS Conditions – if - else

Use the else statement to specify a block of code to be executed if the condition is false

```

if (condition) {
  // code to execute if true;
} else {
  // code to execute if false;
}

```

52

---

---

---

---

---

---

---

---

### JS Conditions – if - else

```

<p>
  Click the button to display<br>
  a time-based greeting:
</p>
<button onclick="myFunction()">
  Try it
</button>
<p id="demo"></p>
<script>
function myFunction() {
  var hour = new Date().getHours();
  var greeting;
  if (hour < 18) {
    greeting = "Good day";
  } else {
    greeting = "Good evening";
  }
  document.getElementById("demo").
  innerHTML = greeting;
}
</script>

```



Click the button to display  
a time-based greeting:



Click the button to display  
a time-based greeting:  
  
Good day

53

---

---

---

---

---

---

---

---

### JS Conditions – else if

Use the else if statement to specify a new condition if the first condition is false

```

if (condition1) {
  // code to execute if condition1 is true;
} else if (condition2) {
  // code to execute if condition1 is false
  // and condition2 is true;
} else {
  // code to execute if both conditions
  // are false;
}

```

54

---

---

---

---

---

---

---

---

### JS Conditions – else if

```

<?php
Click the button to get a time-based greeting:
</?php
<button onclick="myfunction()">
Try it
</button>
<p id="demo"></p>
<script>
function myfunction() {
var greeting;
var time = new Date().getHours();
if (time < 10) {
greeting = "Good morning";
} else if (time < 20) {
greeting = "Good day";
} else {
greeting = "Good evening";
}
document.getElementById("demo").innerHTML = greeting;
}
</script>
    
```

Click the button to get a time-based greeting:

Click the button to get a time-based greeting:   
Good day

East Tennessee State University Department of Computing CSCI 1720 Intermediate Web Design

55

---

---

---

---

---

---

---

---

---

---

### JS Conditions – switch

Use the switch statement to select one of many blocks of code to be executed

Notice that all the cases – except the last one – have break statements. Why?

```

switch (expression) {
case n:
// code to execute if n
break; // exit the statement
case n:
// code to execute if n
break; // exit the statement
case n:
// code to execute if n
break; // exit the statement
case n:
// code to execute if n
break; // exit the statement
case n:
// code to execute if n
}
    
```

East Tennessee State University Department of Computing CSCI 1720 Intermediate Web Design

56

---

---

---

---

---

---

---

---

---

---

### JS Conditions – the break Keyword

When JavaScript reaches a break keyword, it breaks out of the switch block  
This will stop the execution of more code and case testing inside the block  
When a match is found, and the job is done, it's time for a break. There is no need for more testing  
A break can save a lot of execution time because it "ignores" the execution of all the rest of the code in the switch block  
It is not necessary to break the last case in a switch block. The block breaks (ends) there anyway

East Tennessee State University Department of Computing CSCI 1720 Intermediate Web Design

57

---

---

---

---

---

---

---

---

---

---

### JS Conditions – switch

```

<script>
var day;
switch (new Date().getDay()) {
  case 1:
    day = "Monday";
    break;
  case 2:
    day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case 6:
  case 0:
    day = "weekend";
  }
document.getElementById("demo").innerHTML = "Today is " + day;
</script>

```

Today is Tuesday

What's going on here?

East Tennessee State University Department of Computing CSCI 1720 Intermediate Web Design

58

---

---

---

---

---

---

---

---

---

---

### JS Conditions – switch default Case

```

<script>
var text;
switch (new Date().getDay()) {
  case 0:
  case 6:
    text = "It's the weekend!";
    break;
  case 5:
    text = "It's Friday!";
  default:
    text = "Ugh";
}
document.getElementById("demo").innerHTML = text;
</script>

```

Ugh

The default case does not have to be the last case in a switch block

East Tennessee State University Department of Computing CSCI 1720 Intermediate Web Design

59

---

---

---

---

---

---

---

---

---

---

## JavaScript Loops

East Tennessee State University Department of Computing CSCI 1720 Intermediate Web Design

60

---

---

---

---

---

---

---

---

---

---

### JS Loops

Loops can execute a block of code a number of times  
 Loops are handy, if you want to run the same code over and over again, each time with a different value  
 JavaScript supports different kinds of loops:

- for** - loops through a block of code a number of times
- for/in** - loops through the properties of an object
- while** - loops through a block of code while a specified condition is true
- do/while** - also loops through a block of code while a specified condition is true

East Tennessee State University  
 Department of Computing

CSCI 1720  
 Intermediate Web Design

61

---

---

---

---

---

---

---

---

---

---

### JS Loops – for

The for loop is often the tool you will use when you want to create a loop  
 The for loop has the following syntax:

```
for (statement1; statement2; statement3) {
    //code to be executed;
}
```

**Statement 1** is executed before the loop (the code block) starts  
**Statement 2** defines the condition for running the loop (the code block)  
**Statement 3** is executed each time after the loop (the code block) has been executed

```
for (i = 0; i < 5; i++) {
    text += "The number is " + i + "<br>";
}
```

East Tennessee State University  
 Department of Computing

CSCI 1720  
 Intermediate Web Design

62

---

---

---

---

---

---

---


---

---

---

### JS Loops – for

```
<p id="demo"></p>
<script>
var text = "";
var i;
for (i = 0; i < 5; i++) {
    text += "The number is " + i +
    "<br>";
}
document.getElementById("demo").
innerHTML = text;
</script>
```



The number is 0  
 The number is 1  
 The number is 2  
 The number is 3  
 The number is 4

East Tennessee State University  
 Department of Computing

CSCI 1720  
 Intermediate Web Design

63

---

---

---

---

---

---

---

---

---


---

### JS Loops – for/in

The JavaScript for/in statement loops through the properties of an object

```

<ps>
  The for/in statement loops through<br>
  the properties of an object.
</ps>
<p id="demo"></p>
<script>
  var txt = "";
  var person = {fname:"John", lname:"Doe", age:25};
  var x;
  for (x in person) {
    txt += person[x] + " ";
  }
  document.getElementById("demo").innerHTML = txt;
</script>
    
```

The for/in statement loops through the properties of an object.  John Doe 25

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

64

---

---

---

---

---

---

---

---

---

---

### JS Loops – while

Loops can execute a block of code as long as a specified condition is true

The while loop loops through a block of code as long as a specified condition is true

```

while (condition) {
  // code to execute while condition is true;
}
document.getElementById("demo").innerHTML = text;
    
```

**If you forget to increase the variable used in the condition, the loop will never end. This will crash your browser**

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

65

---

---

---

---

---

---

---

---

---

---

### JS Loops – while

The JavaScript for/in statement loops through the properties of an object

```

<p id="demo"></p>
<script>
  var text = "";
  var i = 0;
  while (i < 10) {
    text += "<br>The number is " + i;
    i++;
  }
  document.getElementById("demo").
  innerHTML = text;
</script>
    
```

The number is 0  
The number is 1  
The number is 2  
The number is 3  
The number is 4  
The number is 5  
The number is 6  
The number is 7  
The number is 8  
The number is 9

East Tennessee State University  
Department of Computing

CSCI 1720  
Intermediate Web Design

66

---

---

---

---

---

---

---

---

---

---

### JS Loops – do/while

The do/while loop is a variant of the while loop  
This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true

```
do {
  // code to be executed
  // don't forget to increment
  // the condition
}
while (condition);
```

**If you forget to increase the variable used in the condition, the loop will never end. This will crash your browser**

67

---

---

---

---

---

---

---

---

---

---

### JS Loops – while

The JavaScript for/in statement loops through the properties of an object

```
<p id="demo"></p>
<script>
var text = ""
var i = 0; // condition
do {
  text = "<br>The number is: " + i;
  [i++] // increment condition
}
while (i < 10);
document.getElementById("demo").innerHTML = text;
</script>
```



The number is 0  
The number is 1  
The number is 2  
The number is 3  
The number is 4  
The number is 5  
The number is 6  
The number is 7  
The number is 8  
The number is 9

68

---

---

---

---

---

---

---

---

---

---

### Sources

- [https://www.3schools.com/js/js\\_htmlDOM.asp](https://www.3schools.com/js/js_htmlDOM.asp)
- <https://www.3schools.com/js/default.asp>
- <http://it-ebooks.com/quotes/>
- <https://www.brainyquote.com/quotes/quotes/i/isaacsim100104.html>
- [https://www.goodreads.com/author/quotes/205\\_Robert\\_A\\_Heinlein](https://www.goodreads.com/author/quotes/205_Robert_A_Heinlein)
- <https://en.wikiquote.org/wiki/Dunk>
- <http://www.imdb.com/title/tr0062622/quotes>

69

---

---

---

---

---

---

---

---

---

---

