Lecture 4
JavaScript (Part I)

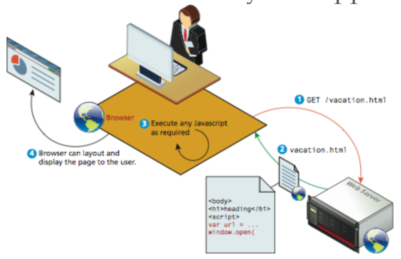
Client/Server Programming
for Internet Applications

TCSS460

Summer 2020

JavaScript: Language Fundamentals

- What is **JavaScript** & What Can it Do?
  - JavaScript is a scripting language which is used to enhance the functionality and appearance of web pages

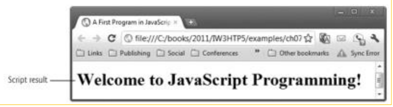


Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

*All major web browsers contain JavaScript **interpreters**, which process the commands written in JavaScript*

```
<!DOCTYPE html>

<!-- Fig. 6.1: welcome.html -->
<!-- Displaying a line of text. -->
<html>
   <head>
      <meta charset = "utf-8">
      <title>A First Program in JavaScript</title>
      <script type = "text/javascript">

         document.writeln(
            "<h1>Welcome to JavaScript Programming!</h1>" );

      </script>
   </head><body></body>
</html>
```

Fig. 6.1 | Displaying a line of text. (Part 1 of 2.)

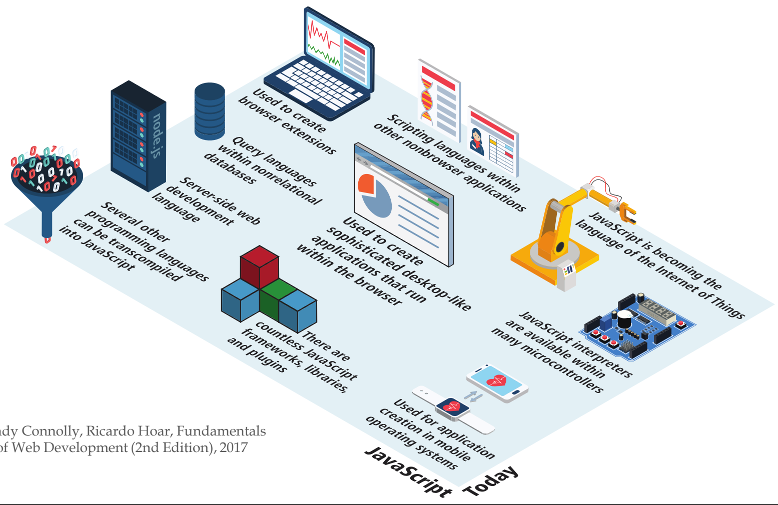


Welcome to JavaScript Programming!

Internet and World Wide Web, Deitel & Deitel , 2012

JavaScript: Language Fundamentals (cont'd)

- What is **JavaScript** & What Can it Do?

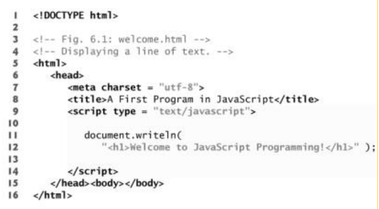

Used to create browser extensions
Scripting languages within other nonbrowser applications
node.js
Query languages within nonrelational databases
Server-side web development language
Several other programming languages can be transcompiled into JavaScript
Used to create sophisticated desktop-like applications that run within the browser
JavaScript is becoming the language of the Internet of Things
There are countless JavaScript frameworks, libraries, and plugins
JavaScript interpreters are available within many microcontrollers
Used for application creation in mobile operating systems
JavaScript Today

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

3

Your First Script: Displaying a Line of Text with JavaScript in a Web Page

- **`<script>`** tag indicates to the browser that the text which follows is part of a script
- **type** attribute specifies the MIME type of the script as well as the scripting language used in the script
  - i.e. a text file written in javascript
  - HTML5: default MIME type for **`<script>`** is "text/html"
  - strings in JavaScript can be enclosed in either
    - **double** quotation marks (") or
    - **single** quotation marks (')
  - **string literal**: string of characters contained between quotation marks



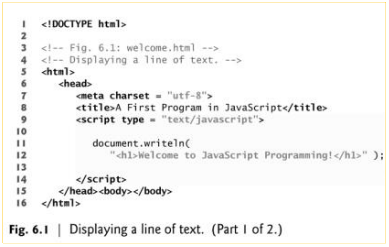
```
<!DOCTYPE html>

<!-- Fig. 6.1: welcome.html -->
<!-- Displaying a line of text. -->
<html>
   <head>
      <meta charset = "utf-8">
      <title>A First Program in JavaScript</title>
      <script type = "text/javascript">

         document.writeln(
            "<h1>Welcome to JavaScript Programming!</h1>" );

      </script>
   </head><body></body>
</html>
```

Fig. 6.1 | Displaying a line of text. (Part 1 of 2.)

4

Your First Script: Displaying a Line of Text with JavaScript in a Web Page

- **document** object represents the HTML5 document currently being displayed in the browser
- browser provides a complete set of objects that enable access and manipulation of every element of an HTML5 document
- object?
  - term generally implies **attributes** (data) and **behaviors** (methods) are associated with the object
    - **methods** → use attributes' data to perform useful actions for the **client** of the object (i.e. script calling the method)



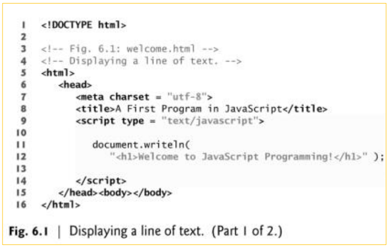
```
<!DOCTYPE html>

<!-- Fig. 6.1: welcome.html -->
<!-- Displaying a line of text. -->
<html>
   <head>
      <meta charset = "utf-8">
      <title>A First Program in JavaScript</title>
      <script type = "text/javascript">

         document.writeln(
            "<h1>Welcome to JavaScript Programming!</h1>" );

      </script>
   </head><body></body>
</html>
```

Fig. 6.1 | Displaying a line of text. (Part 1 of 2.)

5

Your First Script: Displaying a Line of Text with JavaScript in a Web Page

- **parentheses** following the name of a method contain the arguments that the method requires to perform its task (or its action)
  - **document** object's **writeln**
    - writes a line of HTML text in the document



```
<!DOCTYPE html>

<!-- Fig. 6.1: welcome.html -->
<!-- Displaying a line of text. -->
<html>
   <head>
      <meta charset = "utf-8">
      <title>A First Program in JavaScript</title>
      <script type = "text/javascript">

         document.writeln(
            "<h1>Welcome to JavaScript Programming!</h1>" );

      </script>
   </head><body></body>
</html>
```

Fig. 6.1 | Displaying a line of text. (Part 1 of 2.)

- every statement should end with a semicolon (also known as the **statement terminator**), although none is required by JavaScript
- JavaScript is **case sensitive**
  - not using the proper uppercase and lowercase letters is a syntax error

6

Embedding JavaScript Code into HTML5 Documents

- **embedded JavaScript** refers to the practice of placing JavaScript code within a **`<script>`** element

```
<script type="text/javascript">
    /* A JavaScript Comment */
    alert("Hello World!");
</script>
```

- **external JavaScript** files typically contain function definitions, data definitions, and entire frameworks

```
<head>
      <script type="text/javascript" src="greeting.js"></script>
</head>
```

  - makes HTML documents more organized
  - sharing and reuse of .js code among multiple HTML documents

Example 1a

- a script can display **Welcome to JavaScript Programming!** in many ways

```
<!DOCTYPE html>
<html>
<head>
    <meta charset = "utf-8">
    <title>Printing a line with multiple statements</title>
    <script type="text/javascript">
         document.write("<h1 style='color: blue'>");
         document.write("Welcome to JavaScript " +
         "Programming!</h1>");
    </script>
</head>
<body>
</body>
</html>
```

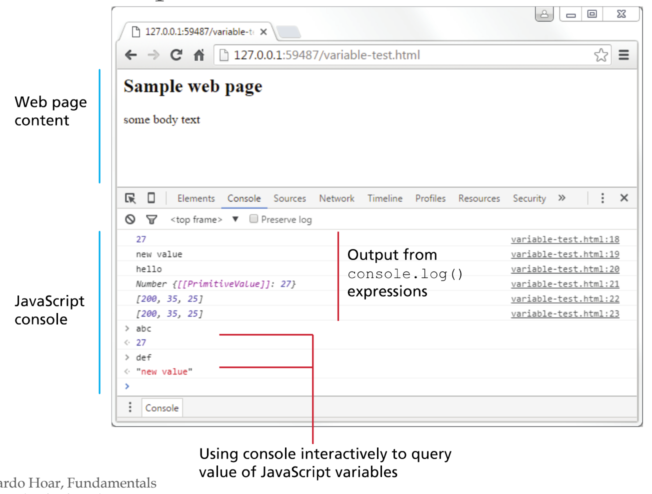
`console.log()` Displays content in the Browser's JavaScript console

concatenation

Welcome to JavaScript Programming!

JavaScript Output

- Chrome JavaScript Console


127.0.0.1:59487/variable-t
127.0.0.1:59487/variable-test.html
Web page content
Elements Console Sources Network Timeline Profiles Resources Security
<top frame>
Preserve log
JavaScript console
Console

Sample web page

some body text

```
27                                      variable-test.html:18
new value                               variable-test.html:19
hello                                   variable-test.html:20
Number {[[PrimitiveValue]]: 27}         variable-test.html:21
[200, 35, 25]                           variable-test.html:22
[200, 35, 25]                           variable-test.html:23
> abc
<· 27
> def
<· "new value"
>
```

Output from `console.log()` expressions

Using console interactively to query value of JavaScript variables

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

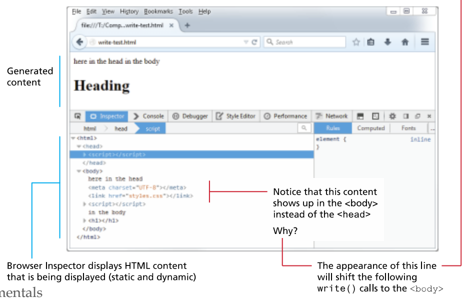
9

JavaScript Output

- Fun with **`document.write()`**

```
<html>
<head>
<script>
    document.write('here in the head');

    document.write('<meta charset="UTF-8">');
    document.write('<link href=styles.css>');
</script>
</head>
<body>
<script>
    document.write("in the body");
    document.write("<h1>Heading</h1>");
</script>
</body>
</html>
```

We want this to appear here in the <head>


File Edit View History Bookmarks Tools Help
file:///T:/Comp...write-test.html
write-test.html
Search
Generated content
Inspector Console Debugger Style Editor Performance Network
html head script
Rules Computed Fonts
element { inline }

here in the head in the body

Heading

```
<html>
  <head>
    <script></script>
  </head>
  <body>
    here in the head
    <meta charset="UTF-8"></meta>
    <link href="styles.css"></link>
    <script></script>
    in the body
    <h1></h1>
  </body>
</html>
```

Notice that this content shows up in the <body> instead of the <head>

Why?

Browser Inspector displays HTML content that is being displayed (static and dynamic)

The appearance of this line will shift the following `write()` calls to the `<body>`

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

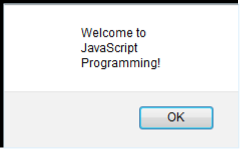
10

Example 1b

- displaying Text in an **Alert** dialog box

```
<!DOCTYPE html>
<html>
<head>
    <meta charset = "utf-8">
    <title>Printing a line with multiple statements</title>
    <script type="text/javascript">
        window.alert("Welcome to \nJavaScript\nProgramming!");
    </script>
</head>
<body>
    <p>Click Refresh (or Reload) to run this script again </p>
</body>
</html>
```

escape sequence (\n for new line)


Welcome to
JavaScript
Programming!

Escape Sequence

Escape sequence	Description
`\n`	*New line*—position the screen cursor at the beginning of the next line.
`\t`	*Horizontal tab*—move the screen cursor to the next tab stop.
`\\`	*Backslash*—used to represent a backslash character in a string.
`\"`	*Double quote*—used to represent a double-quote character in a string contained in double quotes. For example, `window.alert( "\"in double quotes\"" );` displays `"in double quotes"` in an `alert` dialog.
`\'`	*Single quote*—used to represent a single-quote character in a string. For example, `window.alert( '\'in single quotes\'' );` displays `'in single quotes'` in an `alert` dialog.

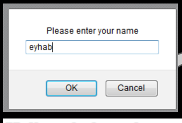
Internet and World Wide Web, Deitel & Deitel , 2012

Example 2

- **Obtaining User Input with prompt Dialogs**
  - gives you the ability to generate part or all of a web page's content at the time it is shown to the user (**dynamic** vs. **static**)
- dynamic welcome page

```
<!DOCTYPE html>
<html>
<head>
      <meta charset = "utf-8">
      <title>Using Prompt and Alert Boxes</title>
      <script type="text/javascript">
      // declare variable called name (commented line...)
      var name;
      name = window.prompt("Please enter your name");
      document.writeln("<h1>Hello " + name + ", welcome to JavaScript Programming!</h1>");
      /*
      this is comment block
      */
      </script>
</head>
<body>
</body>
</html>
```

window object's **prompt** method displays a dialog into which the user can type a value



Hello eyhab, welcome to JavaScript Programming!

Example 3

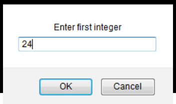
- adding integers

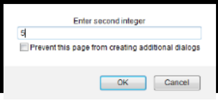
```
<!DOCTYPE html>
<html>
<head>
      <meta charset = "utf-8">
      <title>Addition Example</title>
      <script type="text/javascript">
            var firstNumber, secondNumber, sum;
            // read two integer values from user input

            firstNumber = parseInt(window.prompt("Enter first integer","0"));
            secondNumber = parseInt(window.prompt("Enter second integer","0"));
            sum = firstNumber + secondNumber; // add the two integers

            // display results
            document.writeln("<h1>The sum is " + sum.toString() + ".</h1>");
      </script>
</head>
<body>
</body>
</html>
```

convert numbers from strings to integers

convert from integer to string

if user types non-integer numbers…
`NaN` (not a number): "`The sum is NaN`"

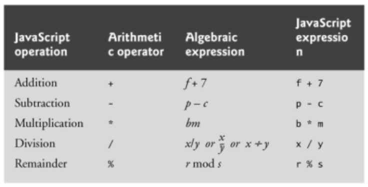




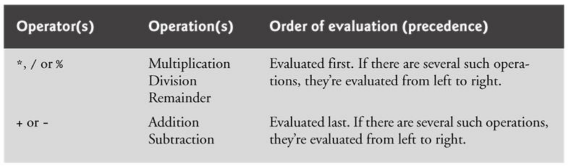
The sum is 29.

Arithmetic

- basic arithmetic operators (+, -, *, /, and %) are binary operators, because they each operate on two operands
- JavaScript provides the remainder operator, %, which yields the remainder after division



JavaScript operation	Arithmetic operator	Algebraic expression	JavaScript expression
Addition	+	$f + 7$	`f + 7`
Subtraction	-	$p - c$	`p - c`
Multiplication	*	bm	`b * m`
Division	/	x/y or $\frac{x}{y}$ or $x \div y$	`x / y`
Remainder	%	$r \bmod s$	`r % s`



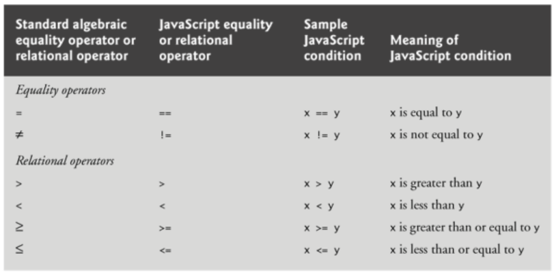
Operator(s)	Operation(s)	Order of evaluation (precedence)
*, / or %	Multiplication Division Remainder	Evaluated first. If there are several such operations, they're evaluated from left to right.
+ or -	Addition Subtraction	Evaluated last. If there are several such operations, they're evaluated from left to right.

Internet and World Wide Web, Deitel & Deitel , 2012

15

Decision Making: Equality and Relational Operators

- **if** statement allows a script to make a decision based on the truth or falsity of a condition
  - if the condition is met (i.e., the condition is true), the statement in the body of the if statement is executed
  - if the condition is not met (i.e., the condition is false), the statement in the body of the if statement is not executed



Standard algebraic equality operator or relational operator	JavaScript equality or relational operator	Sample JavaScript condition	Meaning of JavaScript condition
Equality operators			
=	==	`x == y`	x is equal to y
≠	!=	`x != y`	x is not equal to y
Relational operators			
>	>	`x > y`	x is greater than y
<	<	`x < y`	x is less than y
≥	>=	`x >= y`	x is greater than or equal to y
≤	<=	`x <= y`	x is less than or equal to y

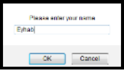
16

Example 4

- display a time-sensitive greeting on a welcome page

```
<!DOCTYPE html>
<html>
<head>
     <meta charset = "utf-8">
     <title>time-related greetings message example</title>
     <script type="text/javascript">
          var name;
          var now = new Date();
          var hour = now.getHours();
          name = window.prompt("Please enter your name");
          if (hour < 12) //morning?
             document.write("<h1>Good Morning, ");
          if (hour > 12) //afternoon?
          {
            hour = hour - 12; // convert to 12-hour clock
            if (hour < 6) // before 6PM?
              document.write("<h1>Good Afternoon, ");

            if (hour >=6) // after 6PM?
              document.write("<h1>Good Evening, ")
          }
          document.writeln(name + ", welcome to JavaScript Programming!</h1>");

     </script>
</head>
<body></body></html>
```

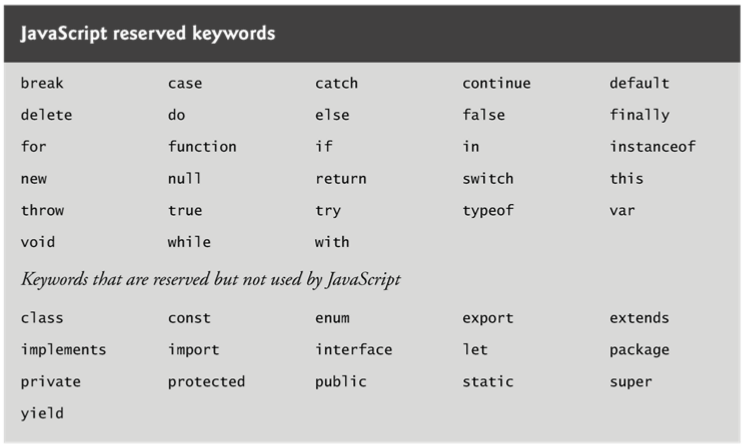
Date object : acquire the current local


Please enter your name
Eyhab

Good Evening, Eyhab, welcome to JavaScript Programming!

17

JavaScript Reserved Keywords



JavaScript reserved keywords

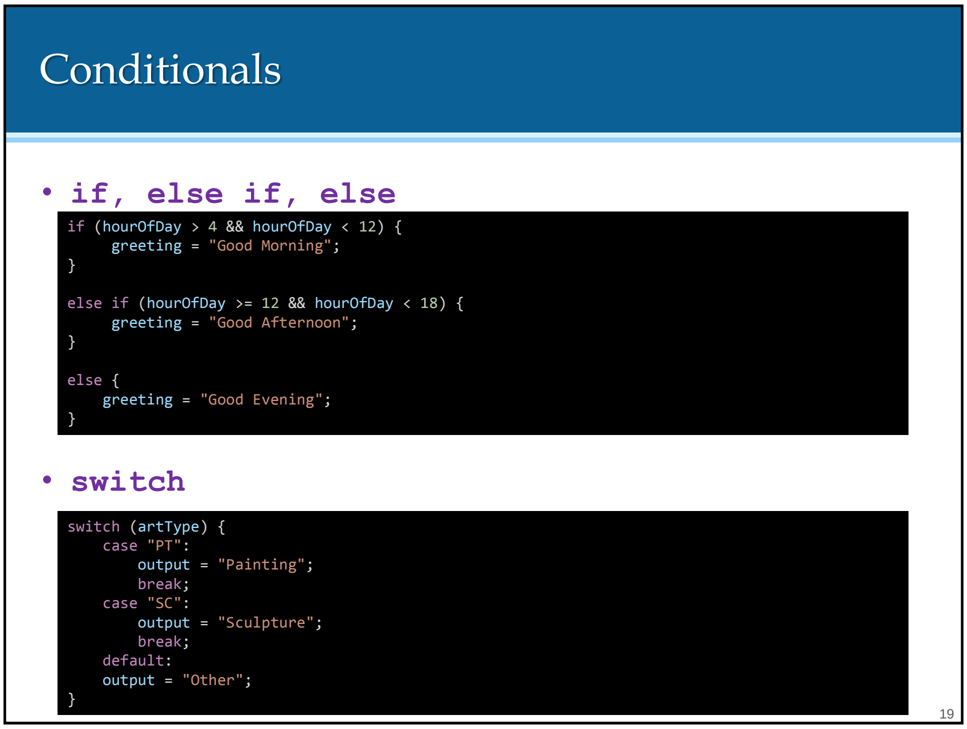
break	case	catch	continue	default
delete	do	else	false	finally
for	function	if	in	instanceof
new	null	return	switch	this
throw	true	try	typeof	var
void	while	with		

Keywords that are reserved but not used by JavaScript

class	const	enum	export	extends
implements	import	interface	let	package
private	protected	public	static	super
yield				

Internet and World Wide Web, Deitel & Deitel , 2012

18



Conditionals

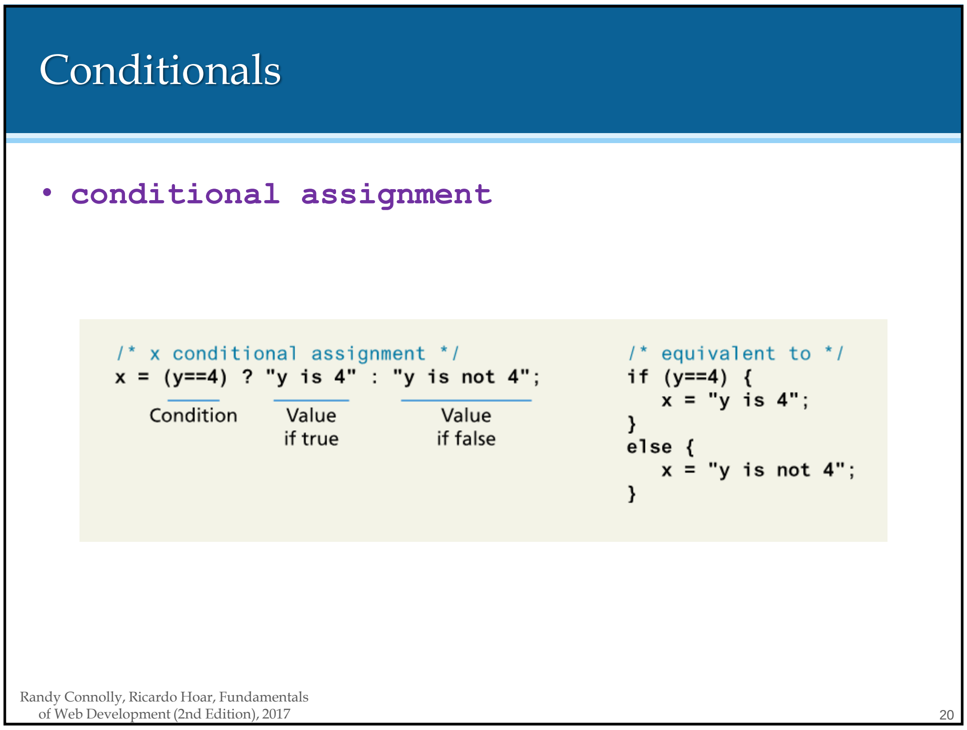
- **if, else if, else**

```
if (hourOfDay > 4 && hourOfDay < 12) {
     greeting = "Good Morning";
}

else if (hourOfDay >= 12 && hourOfDay < 18) {
     greeting = "Good Afternoon";
}

else {
    greeting = "Good Evening";
}
```

- **switch**

```
switch (artType) {
    case "PT":
        output = "Painting";
        break;
    case "SC":
        output = "Sculpture";
        break;
    default:
    output = "Other";
}
```

19



Conditionals

- **conditional assignment**

```
/* x conditional assignment */
x = (y==4) ? "y is 4" : "y is not 4";
```

$$x = \underbrace{(y==4)}_{\text{Condition}}\ ?\ \underbrace{\text{"y is 4"}}_{\text{Value if true}}\ :\ \underbrace{\text{"y is not 4"}}_{\text{Value if false}};$$

```
/* equivalent to */
if (y==4) {
    x = "y is 4";
}
else {
    x = "y is not 4";
}
```

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

20

Loops

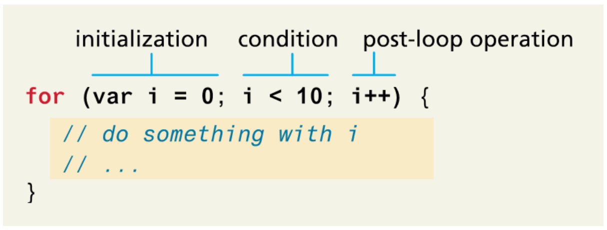
- **while**

```
var count = 0;
while (count < 10) {
    // do something
    // ...
    count++;
}
count = 0;
```

- **do...while**

```
do {
// do something
// ...
count++;
} while (count < 10);
```

- **for**



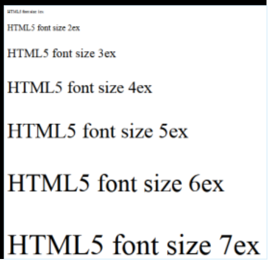
```
for (var i = 0; i < 10; i++) {
    // do something with i
    // ...
}
```

21

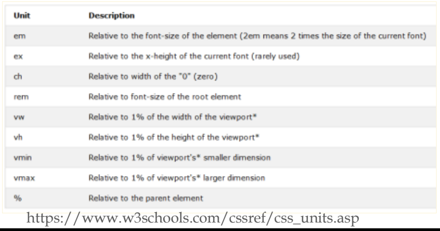
Example 5

- controlling HTML element font-size via a for loop

```
<!DOCTYPE html>
<html>
<head>
      <meta charset = "utf-8">
      <title>for-loop and HTML Example</title>
      <script type="text/javascript">
            for (var counter = 1; counter <= 7; ++counter)
            document.writeln("<p style ='font-size:" +
            counter + "ex'>HTML5 font size " + counter + "ex</p>");
      </script>
</head>
<body>
</body>
</html>
```



```
HTML5 font size 1ex
HTML5 font size 2ex
HTML5 font size 3ex
HTML5 font size 4ex
HTML5 font size 5ex
HTML5 font size 6ex
HTML5 font size 7ex
```



Unit	Description
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ex	Relative to the x-height of the current font (rarely used)
ch	Relative to width of the "0" (zero)
rem	Relative to font-size of the root element
vw	Relative to 1% of the width of the viewport*
vh	Relative to 1% of the height of the viewport*
vmin	Relative to 1% of viewport's* smaller dimension
vmax	Relative to 1% of viewport's* larger dimension
%	Relative to the parent element

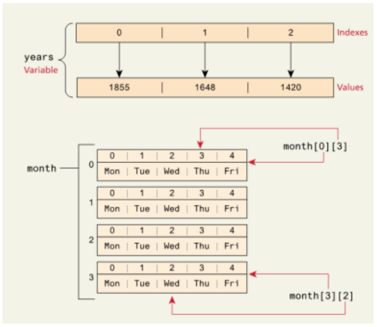
https://www.w3schools.com/cssref/css_units.asp

22

Arrays

```
var years = [1855, 1648, 1420];
var countries = ["Canada", "France","Germany", "Nigeria","Thailand", "United States"];
var mess = [53, "Canada", true, 1420];
```

- some common features
  - arrays in JavaScript are zero indexed
  - [] notation for access
  - `.length` gives the length of the array
  - `.push()`
  - `.pop()`
  - `concat()`, `slice()`, `join()`, `reverse()`, `shift()`, and `sort()`



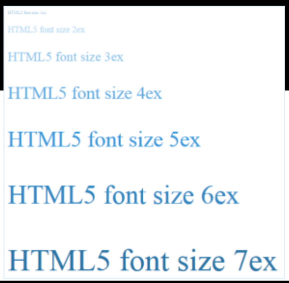
Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

Example 6

- for-loop and array of colors

```
<!DOCTYPE html>
<html>
<head>
      <meta charset = "utf-8">
      <title>for-loop and color example</title>
      <script type="text/javascript">
            var colors = ["d6eaf8", "a9cce3", "aed6f1", "85c1e9",
                          "5dade2", "3498db", "2e86c1", "2874a6"];
            for (var counter = 1; counter <= 7; ++counter)
            document.writeln("<p style ='font-size:" + counter +
            "ex; color: #" + colors[counter] + ";'>HTML5 font size " + counter + "ex</p>");
      </script>
</head>
<body>
</body>
</html>
```

https://htmlcolorcodes.com/



HTML5 font size 1ex

HTML5 font size 2ex

HTML5 font size 3ex

HTML5 font size 4ex

HTML5 font size 5ex

HTML5 font size 6ex

HTML5 font size 7ex

Objects

- **Object Creation – Object Literal Notation**

```
var objName = {
     name1: value1,
     name2: value2,
     // ...
     nameN: valueN
};
```

- **access** using either of:

```
objName.name1
ojName["name1"]
```

- example:

```
// first create an empty object
var objName = new Object();

// then define properties for this object
objName.name1 = value1;
objName.name2 = value2;
```

Functions

- **`functions`** are the building blocks for modular code in JavaScript

```
function subtotal(price, quantity) {
     return price * quantity;
}
```

← Function Declarations

- can be **called** or **invoked** using the () operator

```
var result = subtotal(10,2);
```

- Function Expressions

```
// defines a function using a function expression
var sub = function subtotal(price,quantity) {
    return price * quantity;
};
// invokes the function
var result = sub(10,2);
```

← Function Expressions

Example 7

- Function used for summation

```
<!DOCTYPE html>
<html>
<head>
      <meta charset = "utf-8">
      <title>function, for-loop and summation example</title>
      <script type="text/javascript">
            function sum(min, max, inc)
            {
               var total = 0;
               for (var number = min; number <= max; number +=inc)
                 total += number;

               return total;
            };

            var result = sum(0, 10, 1);

            document.writeln('The sum of integers is ' + result);

      </script>
</head>
<body>
</body>
</html>
```

```
The sum of integers is 55
```

27

Example 8

- Function used for summation (with prompt)

```
<!DOCTYPE html>
<html>
<head>
      <meta charset = "utf-8">
      <title>function, for-loop and summation (with prompt dialog boxes) example</title>
      <script type="text/javascript">
            upper = parseInt(window.prompt("Please enter upper bound"));
            lower = parseInt(window.prompt("Please enter lower bound"));
            increment = parseInt(window.prompt("Please enter increment"));

            function sum(min, max, inc)
            {
               var total = 0;
               for (var number = min; number <= max; number +=inc)
                 total += number;

               return total;
            };

            var result = sum(upper, lower, increment);
            document.writeln('The sum of integers is ' + result);
      </script>
</head>
<body>
</body>
</html>
```

```
The sum of integers is 55
```

28

Functions

- **anonymous function expressions**

```
// defines a function using an anonymous function
expression
var calculateSubtotal = function (price, quantity) {
     return price * quantity;
};
// invokes the function
var result = calculateSubtotal(10,2);
```

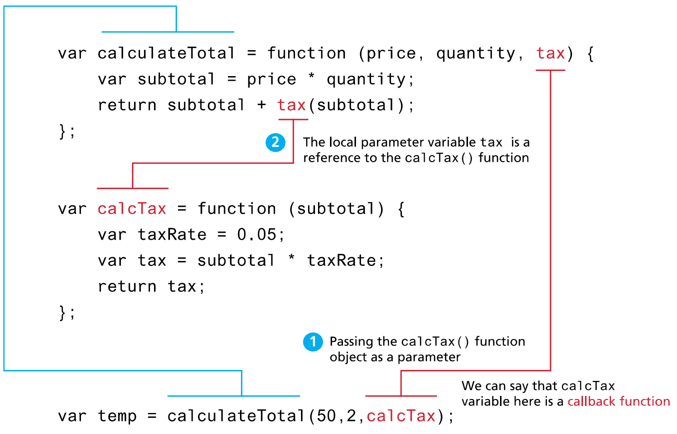
- **nested function**

```
function calculateTotal(price, quantity) {
    var subtotal = price * quantity;
    return subtotal + calculateTax(subtotal);
    // this function is nested
    function calculateTax(subtotal) {
        var taxRate = 0.05;
        var tax = subtotal * taxRate;
        return tax;
    }
  }
```

29

Functions

- **callback Functions**



```
var calculateTotal = function (price, quantity, tax) {
    var subtotal = price * quantity;
    return subtotal + tax(subtotal);
};

var calcTax = function (subtotal) {
    var taxRate = 0.05;
    var tax = subtotal * taxRate;
    return tax;
};

var temp = calculateTotal(50,2,calcTax);
```

2 The local parameter variable tax is a reference to the calcTax() function

1 Passing the calcTax() function object as a parameter

We can say that calcTax variable here is a callback function

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

30

Goals of This Lecture

- You should be comfortable with the basic the JavaScript language fundamentals
- You should be able to know how to add JavaScript into HTML and how a HTML document is represented in JavaScript (i.e. document object)
- You should be comfortable with writing JavaScript code and functions
- You should be able to know how to invoke Console in browsers to review JavaScript output
- You should know how to write conditional statements loops, arrays, and creating objects
- You should know how to create function expressions and callback functions

Module Topics

✓ JavaScript (Part I)

JavaScript (Part II)