Week 2

35

Week 1 Review

In the URL: https://abc.org/contact
org is the _____
https is the _____

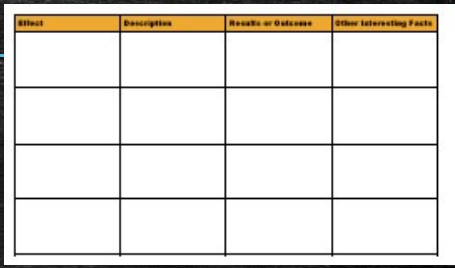
When running an SPA, after accessing the server, the page does not need to _____

In the HTML: <h1 style='text-align: center'>This is my page title</h1>
style is a/an _____

In an HTML list, a bulleted list is created with _____ and each bulleted text is created with _____

36

HTML Tables

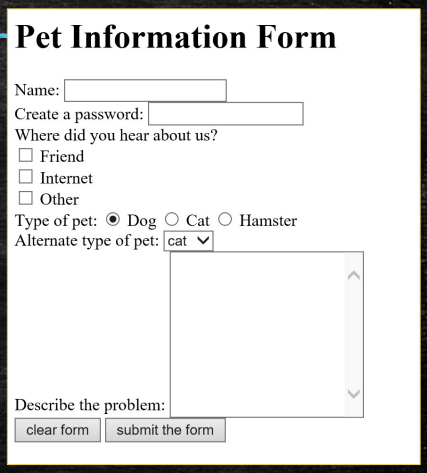


- Create a table: <table>
- Tables are made of rows: <tr>
- Rows are made of elements (table data) <td>
- "Header" rows are bold and centered <th>
- Rows are auto-sized to the tallest item in the row
- Columns are auto-sized to the widest element in the column
- Use   for an empty cell

37

HTML forms

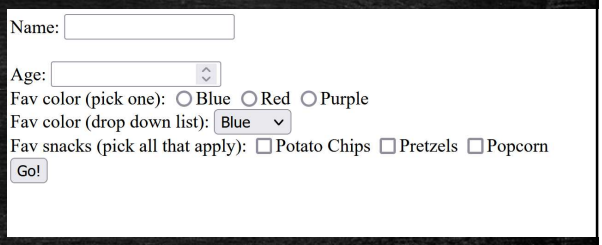
- <form> tag
- <input> tag for text, radio, checkbox, submit, reset
- Radio buttons in the same group all have the same name
- <select> for drop-down/list
  - <option> tag for each item
- <textarea> for multi-line text
  - container tag
  - rows, cols attributes specify number of rows/columns
- Use CSS to align elements
- Use id, name to identify elements for use in script



38

Form Example

```
<form action='#' method='post'>
<label>Name:</label> <input type="text" name='name' id='the_name'><br />
<br />
Age: <input type = "number">
<br />
Fav color (pick one):
        <input type="radio" name="color" value="blue">Blue
        <input type="radio" name="color" value="red">Red
        <input type="radio" name="color" value="purple">Purple
<br />
Fav color (drop down list):
        <select name="favColor">
        <option value="blue">Blue</option>
        <option value="red">Red</option>
        <option value="purple">Purple</option></select>
<br />
Fav snacks (pick all that apply):
        <input type="checkbox" name="chkPotChip" value="potato_chip">Potato Chips
        <input type="checkbox" name="chkPretzel" value="pretzels">Pretzels
        <input type="checkbox" name="chkPopcorn" value="popcorn">Popcorn
<br />
<input type = "button" value = "Go!">
```



39

CSS (Cascading Style Sheets)

- Allows formatting to be separate from content
- Define style instructions through style rules
- Each rule has a selector and a set of property/value pairs

```
selector { style-property1:value;  style-property2:value }
```

- The *selector* indicates what the rule applies to.
- The *properties* are the style characteristics that are being modified.
- The *value* is the new value for that property.
- Multiple property/value pairs are separated by ;

40

Example

```
h1 {text-align:center; color: #ff0000;}
```

- Selector: h1
- Property: `text-align`, Value: `center`
- Property: `color`, Value: `#ff0000`
- This **rule** states that all **h1** tags should be centered on the page and colored red.
- **!important**
  at the end of any property-value pair, `!important` indicates priority
  - `text-align:center !important;`

41

Style rule placement

- **Inline**
  - "at the tag"
  - <p style="color:#ffoooo">
- **Internal: within <style> tag**
  - Less "important" than inline
  - The *closer* a rule is to the selected element, the *stronger* the precedence

```
<style type= 'text/css'>
  h1 {text-align:center;
      color: #ff0000;}
</style>
```

- **External: In a separate file**
  - By convention, style.css
  - No style tag
  - Include on a page using:

<link rel="stylesheet" href="styles.css">

42

Selectors

Type of selector	Example
An HTML tag Applies to any instances of that tag on the page	h1
A style class (starts with a '.') Applies for: class="x" <tag class="x" ...	.my-class
An id (starts with a '#') Applies for: id="x" <div id="x" ...	#the-id
A pseudo-element (starts with a ':') modifies another selector	li:first-child

43

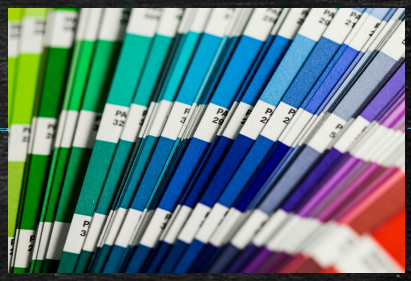
Selector combinations

- a:hover — rule applies when mouse is over the link
- a,b — rule applies to all (note: no space after comma)
- a b — rule applies when b is *contained* within a
  ex: "h1 em" applies for
  "<h1> here is text</h1>"
- tag [attr=value] — rule applies to specified tag only when the attribute is set to the given value
- tag.x — rule applies for tag when its class="x"
  <h1 class="x" ...

44

Colors

- Use hex, rgb or rgba colors
- Hexadecimal colors
  - RGB: red green blue
  - Begin with #
  - #rrggbb
  - Each section ranges from 00 to FF (`#FF0000` is red)
- rgb() color
  - rgb(12, 120, 255)
- rgba() adds transparency
  - rgba(12, 120, 255, .4)



Colors can be used for:

- Text color (color)
- Backgrounds (background-color)
- Border
  - Border-color
  - border: 2px solid #123456;

45

Font & text properties

- font-family
- font-size
- font-style (italic)
- color
- text-align
- text-transform (case)
- text-decoration
- line-height

46

Example

```
<head>
        <title>Demo Page</title>
        <style type="text/css">
                body,html{font-size: 18px}
                p {color: #19A74D}
                p.yellow {color:#D5DE20}
                h1 {
                   font-size:30px;
                   font-family:"Gill Sans", Helvetica, Arial, "sans-serif“
                }
        </style>
</head>
<body>
        <h1>Welcome to my page</h1>
        <h2>This is a subheading</h2>
        <p>And here is some text</p>
        <p class = "yellow">And here is more text</p>
        <p style="color:#ff0000">Want more ideas?</p>
        <a href="https://www.w3schools.com"  target="_blank">
                Try www.w3schools.com
        </a>
</body>
```

47

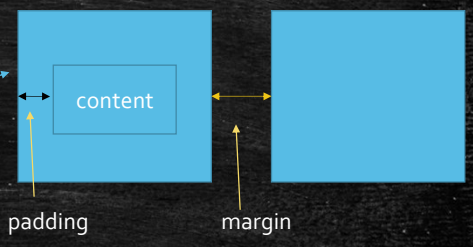
HTML Blocks and the Box Model

Blocks are often used to facilitate styling

- <p>
- Other than line spacing, there is no default styling

The Box Model

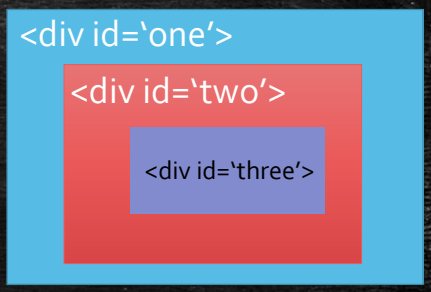
- Border - around an item
- Padding – space between content and boundary of item
- Margin – space between items



48

position

- Absolute
  - Relative to the first parent element with *position* set
- Fixed
  - Relative to the page
  - May also want to use "z-index" property
- Relative
  - Where it would normally go on the page
  - You may need to set position to relative in order to make position absolute work for a child element



one is parent for two
two is parent for three

49

Tricks

- margin or padding: specify up to values - top, left, bottom, right
  - padding: 3px
  - padding: 3px 5px
  - margin: 2px 3px 4px 5px;
- box-sizing: border-box
  - Forces height/width to be specified size
- margin: 0 auto
  - center a block
- position: absolute
  - left: 0; right: 0;
    - centers a block
  - Top: 0; bottom: 0;
    - full column height

50

Example

```
<head>
<title>Blocks Demo</title>
<style type="text/css">
        #top {background-image: url(banner1.jpg);
        background-size:cover;  height: 200px; width: 100%;
        text-align:center; font-size:  40px; padding-top: 100px;
        box-sizing: border-box; }
        .left-col,.right-col {display:inline-block;  width: 49%; margin: 0;}
        .left-col {background-color:  #ccc;}
</style>
</head>
<body>
        <div id='top’> This is the banner </div>
        <div id='content’>
                <div class='left-col’> this goes on the left     </div>
                <div class='right-col’> this goes on the right </div>
        </div> <!-- end content-->
</body>
```

51

JavaScript Programming

52

Programming with JavaScript

- Similar to C/C++
- Scripting language
- Interpreted
- Supports OOP
- Object libraries: DOM (Document Object Model)
- We will be learning ES6, the version used by React
- ES6 => ECMAScript 6 published in 2015

Incorporate into HTML page using <script> tag:

```
<script>
document.write('hello world');
</script>
```

53

53

Input / Output

- We will limit our discussion to elements consistent with what you will encounter in React
- **Output to the page**
  - document.write(), document.writeln()
  - document.write ("hello");
- **Output to the console window**
  - console.log()
- **Write to an element on the page**
  - *The item must be loaded on the page*
  - document.getElementById("result").innerHTML = "hello";
- **Read from a text box (in a form)**

HTML: `<input type='text' id='my-input'>`

JS: `data = document.getElementById('my-input').value;`

54

54

Variables in JavaScript

- Declare a variable
  - Use the keyword "var", "let", "const", or just assign a value.
    - `var x; x = 10;`
    - `let x = 10;`
    - `const x=10;`
    - `x = 10;`
- const cannot be reassigned.
- Scope
  - Global, Local, Block

- Scope of var vs let vs const
  - var is function scoped
  - let and const are block scoped
- Declare several variables at the same time
  - `var x, y;`
  - `let x = 10, y;`
- *null* assigns a non-value to a variable
  - var x=null;

55

55

Scope Demo

```
<script>
        var globalScope=0;
        function foo() {
         if (true) {
                var funcScope = 1;
                let blockScope = 2;
                console.log(funcScope);        // will print 1
                console.log(blockScope);       // will print 2

                let globalScope = 3;
                if (true) {
                 console.log(funcScope);       // will print 1
                 console.log(blockScope);      // will print 2
                 console.log(globalScope);     // will print 3
                }
         }
         console.log(funcScope);               // will print 1
         console.log(globalScope);             // will print 0
         console.log(blockScope);              // error
        }
        foo();
</script>
```

56

Data Types in JavaScript

- JavaScript is loosely typed- data types are deduced.
  (Variables in JavaScript are shape shifters!)
- Numbers
  - Integral (no decimal point)
  - Floating point (*may* have a decimal point)
- Text (Strings)
  - Enclosed in quotes
  - Both single and double quotes are valid (end with what you start with)
- Boolean
  - True or false

57

57

Working with Data

- parseInt converts a string to a number
- The converted value is returned - the original value is not changed
- parseInt vs parseFloat
- NaN (not a number)
- Note: the result of prompt and reading a form is always a string

```
alert(parseInt("1 is the loneliest number"));
    //result is 1

alert(parseInt("The loneliest number is 1"));
    // result is NaN

alert(parseInt("3.14 is my favorite kind of pi"));
    // result is 3

alert(parseFloat("3.14 is my favorite kind of pi"));
    // result is 3.14
```

58

58

Strings and Numbers

- Use + to concatenate (glue) strings
  - x = "hello "; y = " there";
    z = x + y; // z is "hello there"
- If you concatenate a string with a number, the result is a string
  - x = "hello "; y = "12"; z = x + y; // z is "hello 12"
  - n = 4; n = n + ""; // n is "4"
  - n = 4; z = "The answer is " + n + 2); // z is "The answer is 42"
  - n =4; z = "The answer is " + (n + 2)); // z is The answer is 6"

59

59

Try it

Create an HTML page as follows:

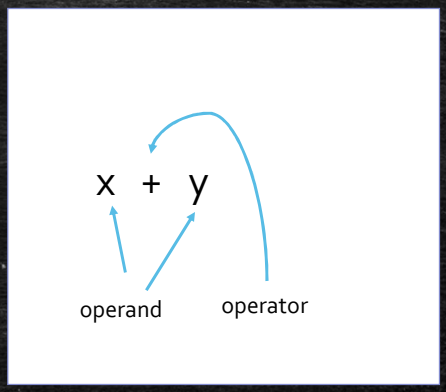
- Add an h1 to the page that displays: "My JavaScript Page"
- In a script section:
  - Create a variable called name using var.
  - Set the value of the variable to your name
  - Display the name on the page using document.write
  - Display the name in the console using console.log
  - Create a variable called age using let, and set it to 42
  - Display the name and the age in 2 years using document.write

60

60

Operators

- Arithmetic
- Arithmetic with assignment
- Assignment
- Comparison
- Logical
- Concatenation



61

61

Arithmetic

+	addition	Adds numeric operands.
-	subtraction	Used for negating or subtracting.
++	increment	Add 1 to the operand.
--	decrement	Subtract 1 from the operand.
*	multiplication	Multiplies two numerical operands.
/	division	Divides first operand by second operand
%	modulus	Calculates the remainder of first operand divided by second operand.

62

62

Operator Precedence

- JS follows PEMDAS
- "Please Excuse My Dear Aunt Sally"
- Parentheses
- Exponent
- Multiplication, Division
- Addition, Subtraction
- See: https://www.dummies.com/article/technology/programming-web-design/general-programming-web-design/javascript-operator-precedence-254119

63

63

Example: Tip Calculator

```
var checkAmount, tipAmount;
const TIP_PERCENT=.17;
checkAmount = prompt("What is the check amount? ","20");
document.write("Check amount: $"
                + checkAmount
                + "Tip: $ "
                + checkAmount * TIP_PERCENT);

// why isn't parseInt needed?
```

64

64

Example: Is a number even?

```
var num, result;
num = prompt("Enter a number", "5");
result = num % 2;
document.write ("The remainder of " + num + “divided by 2 is " + result);
```

65

65

Compound Assignment Operators

- +=, -=, *=, /=, %=
- Performs the operation and does the assignment
- Example:
  `x = x + 3;`
  is the same as
  `x += 3;`

Four ways to add 1 to a value

```
x = x+ 1;
x++;
++x;
x += 1;
```

66

66

Comparison

== equality	True if two operands have the same value (data conversion may occur)
!= inequality	Opposite of the equality (==) operator.
> greater than	True if first operand is larger
>= greater or equal	True if first operand is greater than or equal to a second operand
< less than	True if first operand is smaller
<= less than or equal	True if first operand is less than or equal to a second operand

67

67

Conditional Operator ?:

- test ? result1 : result2
- Do the test, if the result is true, then the value of the operator is Result1 otherwise it is Result2
- Remember: this is an operator, there is no implied assignment
- "Absolute value" example:
  - `n = n<0 ? -n : n;`

68

68

Example: Even Revisited

```
var num, result;
num = prompt("Enter a number", "5");
result = num % 2;
document.write ("The number: " + num
           +  (result==0 ? " is " : " is not ")
           + "even";
```

69

69

Logic Operations

- Logical operands "glue" two comparison operators together.
- JavaScript supports three logical operations:
  - && AND (shift-7) true when two operands are **both** true
  - || OR (shift \) true when **either** of two operands is true
  - ! NOT (shift 1) (opposite) true when an operand is false
- For example, there is no "between" operator, but you can accomplish the same thing with AND

```
between = z>10 && z<=20;   // true when z is 11 through 20 inclusive
```

70

70

Logic Operations (continued)

- When a value of one input expression forces the output result of a logical operator, it is called a "Forcing Function"
  - The forcing function for AND is false.
  - The forcing function for OR is true.
- Shortcut calculation- if there are two parts to an expression and the first part forces the value, the second part is not evaluated

```
x = (6<3 )&& (7<4);  // no need to evaluate (7<4)
```

71

71

Mixing Numbers, Strings and Operators

- + operator
  - string + string — concatenation
  - string + number — convert number to string and then concatenate
  - number + number — addition
- Comparison operators (ex, == >)
  - string > string — alphabetical order
  - string > number — convert string to number and compare numerically
  - number > number — numerical order

72

72

Figure it Out

- What is:

  `3 > 4 && 6 < 10`

  `“6” == 6 || 8< 0`

  `! (5 > 2)`

- Assume n = 10 for each what is:

  `n > 0`

  `n <= 5 && n >= 10`

  `n >= 1 && n<= 10`

  `n == “10” || n < 3`

73

73

Prefix vs Postfix Notation

- x++
  value is x
  add one to x
- ++x
  value is x+1
  add one to x

Given, x = 3

```
y = x++;     //y = 3,  x = 4
y = ++x;     //y = 5,  x = 5
```

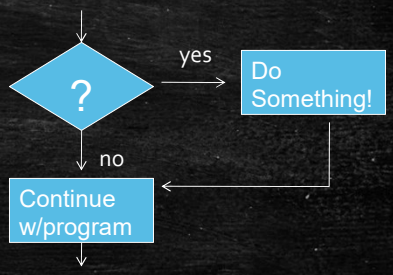
74

74

Conditionals

- The IF construct helps determine PROGRAM FLOW based on a yes/no question.

- `if (n>1)`
  `alert ("Do something");`



75

75

Simple IF statement

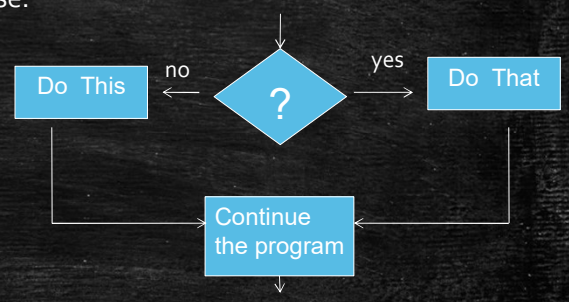
- `if (n>=0)`
  `alert("N is a positive number");`
- Following the keyword "if" is a *conditional expression* in parenthesis - an expression that is evaluated as true of false
  - Comparison operators
  - Boolean logic operators (and, or)
  - Anything can be evaluated as true/false (recall that zero is false)
- If the expression is true, then do the statement immediately following the "if"
  Otherwise, skip that statement.
  - Several statements can be included in an "if" block by enclosing them in a code block { . . . }

76

76

Otherwise …

- Sometimes we want to take different actions when the conditional expression is true vs when it is false.
  - Use an ELSE statement (think "otherwise"_
- ```
  if (n>=0)
      alert("n is a positive number");
  else
      alert("n is a negative number");
  ```
- You can do several statements after the else by enclosing them in a code block { . . . }



77

77

Notes

- Each part of an if / else if/ else statement is mutually exclusive. i.e., only one can be true.
- *Do not* put a conditional expression after a lone "else"
  ie: else (n>3) //WRONG!
- You must use curly brackets when there is more than one statement after an if or else. Curly brackets are optional for a single statement.

78

78
  ```

*Which code has a **syntax ERROR:***

```
a. if (a<b)
   document.write( "hello" );

b. if a<b
   document.write( "hello" );

c. if (a<b) && (b<c)
  {
    document.write( "hello" );
    document.write( "there" );
  }

d. if (a<b) && (b<c)
   document.write( "hello" );
   document.write( "there" );

e. if (a=4)
     document.write( "hello" );
```

79

79

Examples: Solve using if/else statements

- If a guess matches a number, display "You are correct", otherwise display "Sorry, try again"

```
if (guess==number)
        alert("correct");
else
        alert("Sorry, try again");
```

- If a guess matches a number, display "You are correct", otherwise display "Try again – you may have three guesses" and add one to a variable called numGuesses.

```
if (guess==number)
        alert("correct");
else
{
        alert("Try again – you may have three guesses");
        numGuesses+= 1;
}
```

80

80

Switch – shorthand for IF

- ```
  switch(expression)
  {
     case value1: do this; break;
     case value2: do that; break;
     default: do the other thing; break;
  }
  ```
- *break* keeps it from "falling through"
- *default* placed at the end and is optional

81

81

Problem: solve the following using switch

- You have a variable called coin and a variable called value.
- If coin is "nickel", value is 5
- If coin is "dime", value is 10
- If coin is "quarter" value is 25
- Create a switch statement that determines the value of the coin

```
switch(coin)
{
   case "nickel" :
      val = 5;
      break;
   case "dime" :
      val = 10;
      break;
   case "quarter" :
      val = 25;
}
```

82

82
```

Figure it out

- What is the value of x at the end?

```
y = 5;  x= 7;
switch(y)
{
  case 1:   x= x*2; break;
  case 5:   x -= y;
  case 10:  x++; break;
  default:  x = 17;
}
```

83

83

Repetition using Loops

- **Two types of loops: Counting and Waiting**
  - In a **counting loop**, you do something for a specified number of times.
    - For example – display "Hello World" on the screen 10 times.
    - Or maybe, move two squares – 4 times.
    - *In JavaScript, a **for** loop is optimized for counting*
  - In a **waiting loop**, you do something until something happens.
    - For example – get numbers from the user until the user enters: -1
    - Or, move a robot forward 1 inch at a time–until a boundary is detected.
    - *In JavaScript, a **while** loop is optimized for waiting.*

84

84

FOR Loop Syntax

```
for (init ; test ; update)
{
  // loop statement(s)
}
```

Display the numbers: 1 to 10

```
for (n=1; n<=10; n++)
  alert(n);
```

Notes:

- **initialization**- statement that occurs prior to any iteration
- **test** - conditional expression that is evaluated at the beginning of each iteration. When expression evaluates to false, exit the loop
- **update** - statement that occurs at the end of each iteration. Often used to update a counter.
- *Any or all of these can be omitted*
  - for (;;) is an intentional infinite loop
- Do not put a semicolon at the end of a "for" header

85

85

WHILE loop syntax

- Continue to *iterate* as long as the conditional expression is true
- Alternate form: do-while allows for at least one iteration through the loop

```
while (test)
{
  // loop statement(s)
}

do  {
  // loop statement(s)
} while (test);
```

86

86

Example: display the numbers from 1 to 10

```
n = 1;              // initialization
while (n<=10)       // test
{
  alert(n);
  n++;              // update
}
```

Notes:

- Do not put a semicolon at the end of a while header *except* in a do-while construct.
- while(true) is an intentional infinite loop

87

87

Week 3

88

Week 2 Review

A CSS style rule consists of _____, _____ and _____

An internal stylesheet uses the _____ tag.

The CSS selector: UL,LI will apply to _____
What if it is changed to: UL LI _____

position: absolute places an element relative to _____

JavaScript code is executed at the _____

I want to create a variable called "n" within an if statement that will only be accessible within the if statement – how can I make this happen?

In the expression: `exp1 ? result1 : result2;`
exp1 is a/an _____

89

Figure it Out

How many times will "hello" be printed?

```
for (i=1; i<=5; i++)
  document.write( "hello" );
while (i<= 5)
  document.write( "hello" );
```

How many times will "hello" be printed?

```
for (i=2; i<=5; i++);
  document.write( "hello" );
```

90

90

Break And Continue

- Break and continue statements provide additional control for directing the flow through a loop.
- **Break-** exits a loop.
  - Generally used as part of an *if* construct.
  - Used to provide an alternate place to exit from the loop
  - Use carefully to avoid unreadable code.
- **Continue-** ends the current iteration of a loop.
  - In a while loop, continue goes directly to the *test*
  - In a for loop, continue goes directly to the *update*.

91

91

Break and Continue

```
for (i=1; i<=5; i++)
{
  if (i==2) continue;
  document.write( "hello"
  );
  if (i==4) break;
}
```

```
i=0;
while (i<=5)
{
  if (i==2) continue;
  document.write( "hello"
  );
  if (i==4) break;
}
```

92

Example

```
//this loop demonstrates two exit points
  while(true)
  {
       if (x == 10)  break;          //get out here
       if (x == 20)  break;          //or get out here
       x++;
  }
// special case for first iteration
  for (x=0; x<10; x++)
  {
             document.write(x);
             if (x==0)
                    continue;
       // more loop statements can go here
  }
```

93

93

Try It

- Solve the following with a for loop and then with a while loop:
  - Count down by two's from 40 to 10. i.e., 40,38,36,34,… (display each number).
- Solve the following using a while loop
  - Use prompt to get numbers from the user until the sum of the numbers exceeds 20

94

Functions

- A function is a named set of tasks that are not executed until the function is *called*
- Functions can be *anonymous* (no name) and are used as a parameter to another function.
- Functions can have inputs, outputs, and byproducts
- Use the *return* statement to end the function or return a value

```
function add1(p1,p2)
{
  var a = p1, b = p2, sum;
  sum = a + b;
  alert ("The sum is: " + sum);
}

add(3, 4);
```

95

95

The return statement

```
function add2()
{
  var a = 2, b = 3, sum;
  sum = a + b;
  return sum;
}
```

Call the function

```
var sum;
sum = add2();
alert("The sum is: " + sum);
```

96

96

Providing Arguments (Parameters)

```
function add3(a,b)
{
  var sum = a + b;
  alert(sum)
}
```

Calling the function

```
add3(2,3);
```

97

97