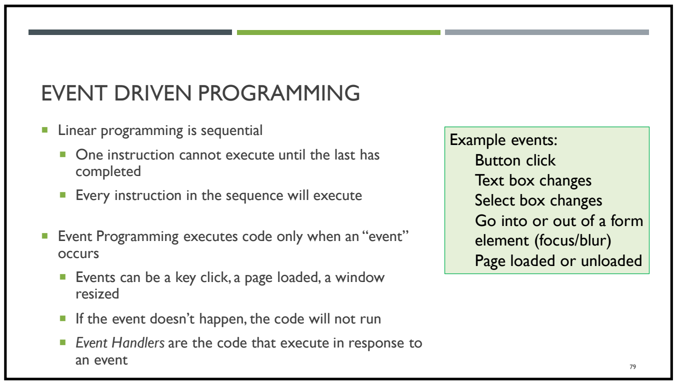
COMP 20 WEB PROGRAMMING

WEEK 6

78

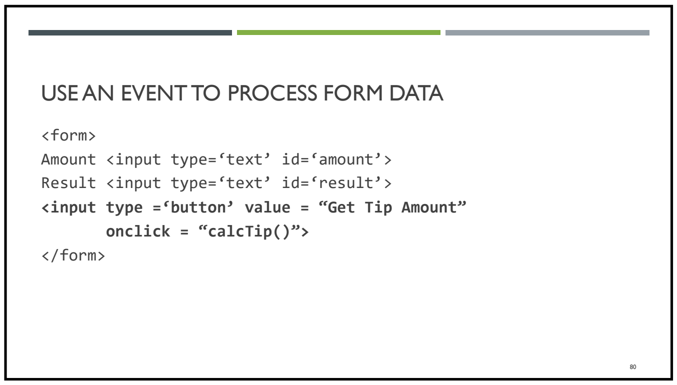


EVENT DRIVEN PROGRAMMING

- Linear programming is sequential
  - One instruction cannot execute until the last has completed
  - Every instruction in the sequence will execute
- Event Programming executes code only when an "event" occurs
  - Events can be a key click, a page loaded, a window resized
  - If the event doesn't happen, the code will not run
  - *Event Handlers* are the code that execute in response to an event

Example events:
- Button click
- Text box changes
- Select box changes
- Go into or out of a form element (focus/blur)
- Page loaded or unloaded

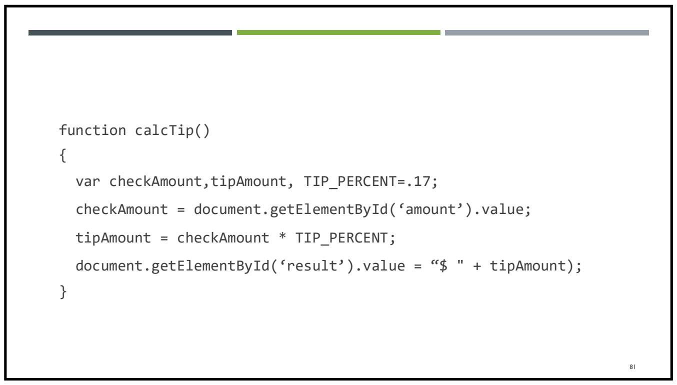
79



USE AN EVENT TO PROCESS FORM DATA

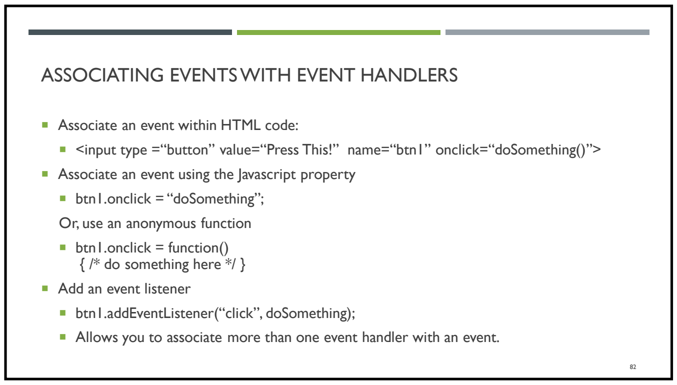
```
<form>
Amount <input type='text' id='amount'>
Result <input type='text' id='result'>
<input type ='button' value = "Get Tip Amount"
       onclick = "calcTip()">
</form>
```

80



```
function calcTip()
{
  var checkAmount,tipAmount, TIP_PERCENT=.17;
  checkAmount = document.getElementById('amount').value;
  tipAmount = checkAmount * TIP_PERCENT;
  document.getElementById('result').value = "$ " + tipAmount);
}
```

81



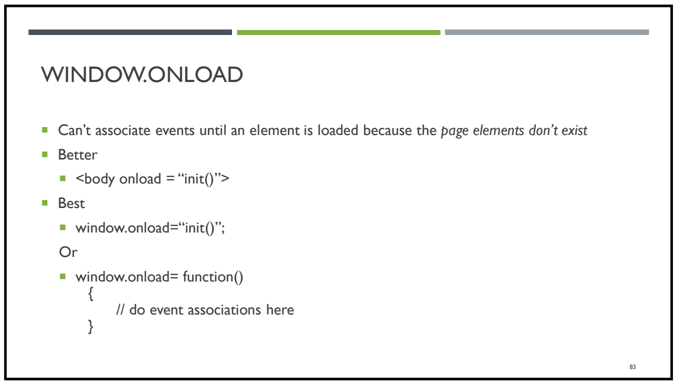
ASSOCIATING EVENTS WITH EVENT HANDLERS

- Associate an event within HTML code:
  - <input type ="button" value="Press This!" name="btn1" onclick="doSomething()">
- Associate an event using the Javascript property
  - btn1.onclick = "doSomething";

  Or, use an anonymous function
  - btn1.onclick = function()
    { /* do something here */ }
- Add an event listener
  - btn1.addEventListener("click", doSomething);
  - Allows you to associate more than one event handler with an event.

82

82



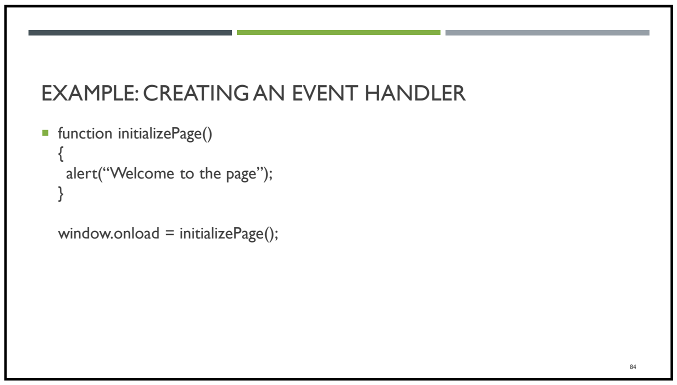
WINDOW.ONLOAD

- Can't associate events until an element is loaded because the *page elements don't exist*
- Better
  - <body onload = "init()">
- Best
  - window.onload="init()";

  Or
  - window.onload= function()
```
    {
        // do event associations here
    }
```

83

83



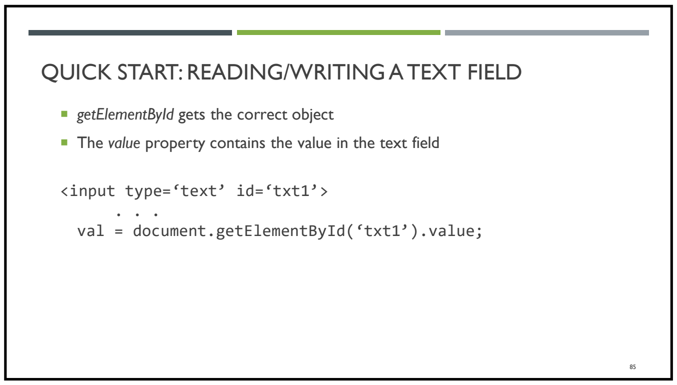
EXAMPLE: CREATING AN EVENT HANDLER

- function initializePage()
```
{
  alert("Welcome to the page");
}

window.onload = initializePage();
```

84

84



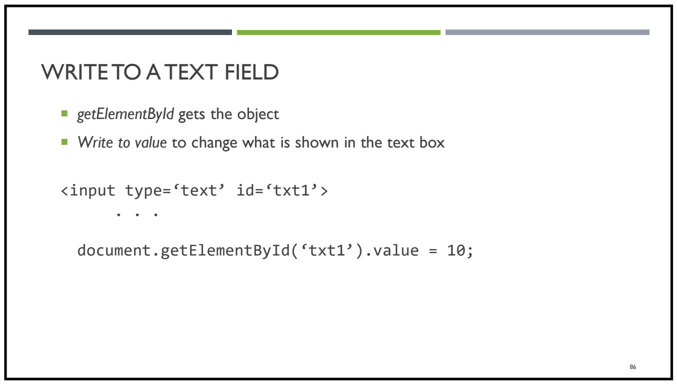
QUICK START: READING/WRITING A TEXT FIELD

- *getElementById* gets the correct object
- The *value* property contains the value in the text field

```
<input type='text' id='txt1'>
      . . .
  val = document.getElementById('txt1').value;
```

85

85



WRITE TO A TEXT FIELD

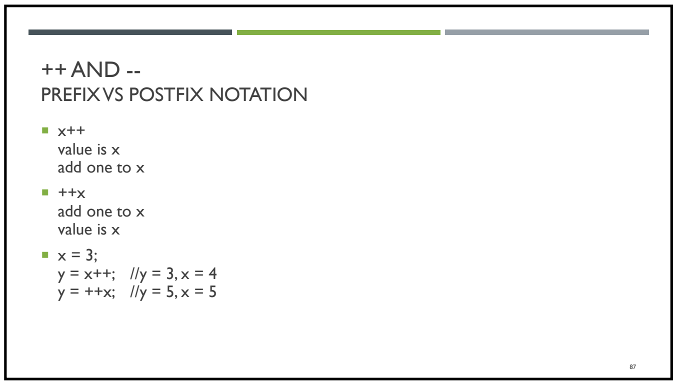
- *getElementById* gets the object
- *Write to value* to change what is shown in the text box

```
<input type='text' id='txt1'>
      . . .

  document.getElementById('txt1').value = 10;
```

86

86



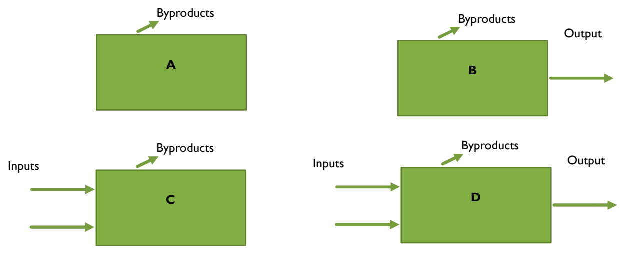
++ AND --
PREFIX VS POSTFIX NOTATION

- x++
  value is x
  add one to x
- ++x
  add one to x
  value is x
- x = 3;
  y = x++;   //y = 3, x = 4
  y = ++x;   //y = 5, x = 5

87

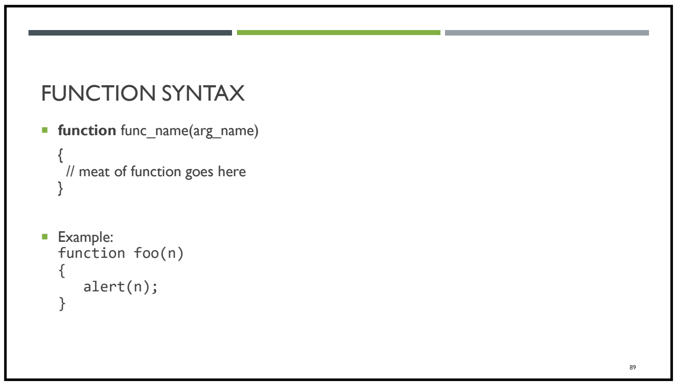
87

BLACK BOX MODEL FOR FUNCTIONS



88

88



FUNCTION SYNTAX

- **function** func_name(arg_name)
  {
    // meat of function goes here
  }
- Example:
```
function foo(n)
{
   alert(n);
}
```

89

89

EXAMPLES

Function definition (Type A)

```
function add1()
{
var a = 2, b = 3, sum;
  sum = a + b;
  alert ("The sum is: " +
       sum);
}
```

Calling the function

```
add1();
```

90

GIVING BACK A VALUE - RETURN

Function definition (Type B)

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

Calling the function

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

91

PROVIDING ARGUMENTS (PARAMETERS)

Function definition (Type D)

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

Calling the function

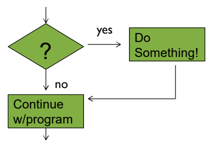
```
var sum;
sum = add3(2,3);
alert ("The sum is: "
    + sum);
```

92

DECISIONS, DECISIONS …

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

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



93

SIMPLE IF STATEMENT

-
  ```
  if (n>=0)
   alert("N is a positive number");
  ```
- After the keyword "if" is a *conditional expression* - an expression that results in a true/false answer
  - 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 executed by enclosing them in a code block {…}

94

PROBLEM: SOLVE THE FOLLOWING USING IF STATEMENTS

- If a variable named "likesBeer" has a value of true, display "I like beer too!"
- If a variable called "numPets" is 0, display "you do not have pets"
- Compare two numbers, n1 and n2. If the first number is greater than the second display "The first number is bigger"
- Compare two numbers, n1 and n2. If the first number is greater than the second display "The first number is bigger" and add 10 to the first number.

95

SOLUTIONS

-
  ```
  if (likesBeer)
   alert("I like beer too!");
  ```

-
  ```
  if (!numPets)
   alert("You do not have pets");
  ```

-
  ```
  if (n1 > n2)
   alert( "The first number is bigger" );
  ```

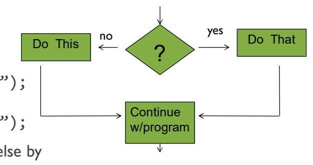
-
  ```
  if (n1 > n2)
  {
   alert( "The first number is bigger" );
   n1+= 10;
  }
  ```

96

OTHERWISE …

- Sometimes we want to take action whether or not the conditional expression is true.
  - To express "otherwise", use an ELSE statement.

-
  ```
  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 {…}



97

PROBLEM: SOLVE THE FOLLOWING USING IF/ELSE STATEMENTS

- If a guess matches a number, display "You are correct", otherwise display "Sorry, try again"
- If a variable called drinksMilk is true, display "Got milk?". Otherwise, display "Milk is good for you".
- Compare two numbers, n1 and n2. If the first number is greater than the second display "The first number is bigger" and add 10 to the first number, otherwise display "The second number is bigger" and add 10 to the second number.

98

98

SOLUTIONS

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

```
if (drinksMilk==true)
  alert("Got milk?");
else
  alert("Milk is good for you");
```

```
if (n1 > n2)
{
 alert( "The first number is
bigger" );
 n1+= 10;
}
else
{
 alert( "The second number is
bigger" );
 n2+= 10;
}
```

99

99

QUESTIONS, QUESTIONS …

- Use "else if" to keep asking questions

```
if (mm<4)
 alert("You have a few m&m's");
else if (mm< 10)
     alert("You have a handful of m&m's");
else if (mm< 30)
     alert("You have several m&m's");
else
{
  alert("You have a lot of m&m's");
  alert("Can I have some?");
}
```

100

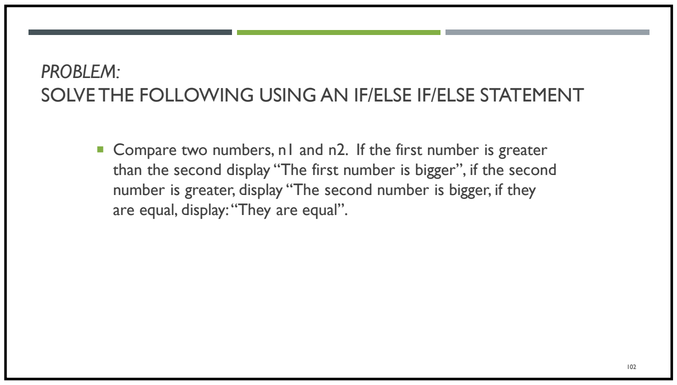
100

NOTES

- Parts of an if / else if/ else statement are mutually exclusive. i.e., only one can be true.
- *Do not* put a conditional expression after a lone "else"
- Always use curly brackets when there is more than one statement after an if or else (and its ok to use curly brackets for even one statement in a code block)

101

101

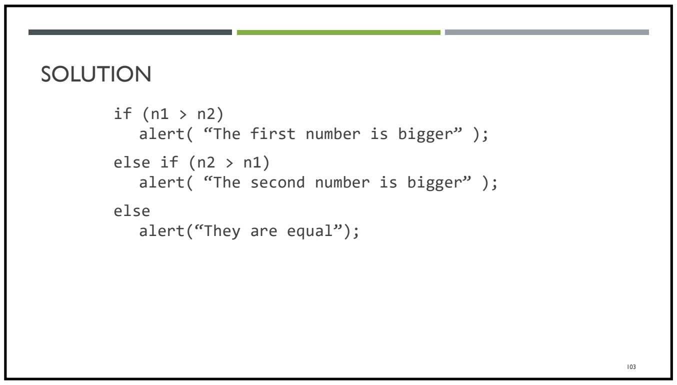


PROBLEM:
SOLVE THE FOLLOWING USING AN IF/ELSE IF/ELSE STATEMENT

- Compare two numbers, n1 and n2. If the first number is greater than the second display "The first number is bigger", if the second number is greater, display "The second number is bigger, if they are equal, display: "They are equal".

102

102

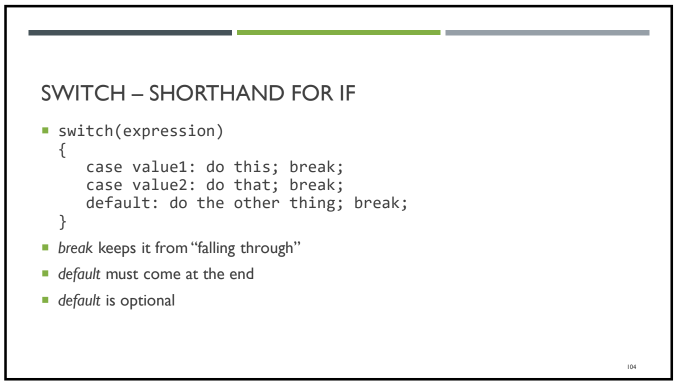


SOLUTION

```
if (n1 > n2)
   alert( “The first number is bigger” );
else if (n2 > n1)
   alert( “The second number is bigger” );
else
   alert(“They are equal”);
```

103

103

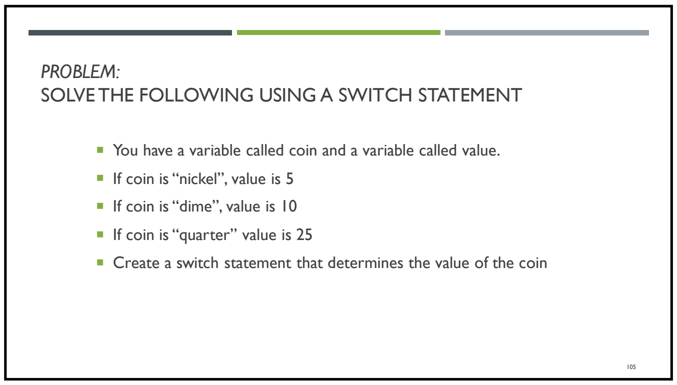


SWITCH – SHORTHAND FOR IF

-
  ```
  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* must come at the end
- *default* is optional

104

104



PROBLEM:
SOLVE THE FOLLOWING USING A SWITCH STATEMENT

- 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

105

105

SOLUTION

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

LOOPS: WHEN ONCE IS NOT ENOUGH!

- A loop is when your program does something over, and over, and over and over …
- 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 a **waiting loop**, you do something until something else 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 counting loop is done with **for** and a waiting loop is done with **while**.*

FOR LOOP

- Three parts built-in to the header:
  - **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.*

GENERAL FORMAT

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

Display the numbers: 1 to 10

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

Notes:
- Do not put a semicolon at the end of a “for” header
- for (;;) is an intentional infinite loop
- Omit the initialization and update segments to emulate a *while* loop.

TRY IT: SOLVE USING A FOR LOOP

- Display the even numbers from 10 to 40
- Count down from 10 to 1

110

110

WHILE LOOP

- 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);
```

111

111

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

112

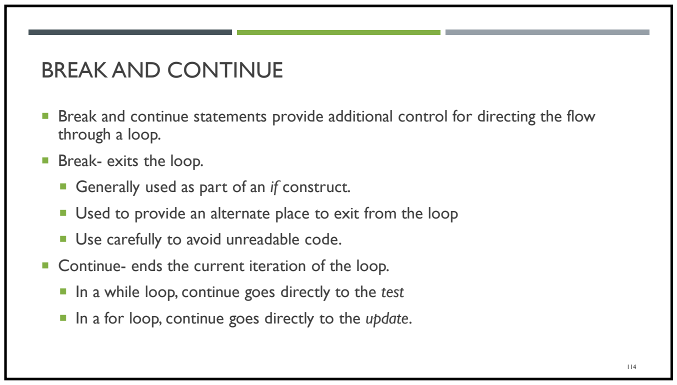
112

TRY IT: SOLVE THE FOLLOWING USING A WHILE LOOP

- Display “hello” on the screen 5 times
- Get random numbers between 1 and 15 until one of the numbers is greater than 10. Display the sum of the numbers.

113

113

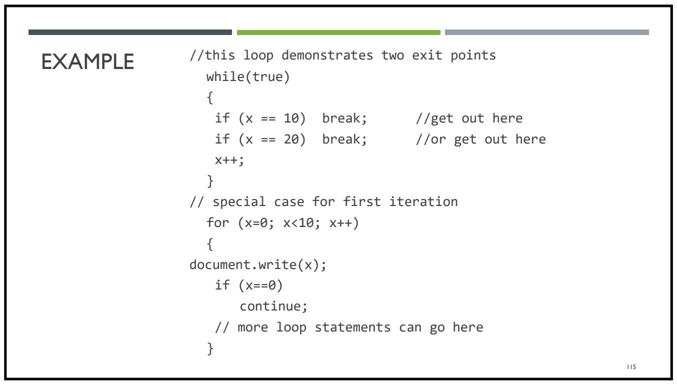


BREAK AND CONTINUE

- Break and continue statements provide additional control for directing the flow through a loop.
- Break- exits the 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 the loop.
  - In a while loop, continue goes directly to the *test*
  - In a for loop, continue goes directly to the *update*.

114

114

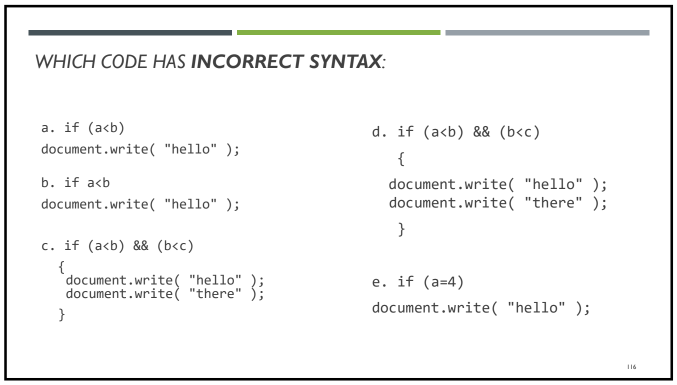


EXAMPLE

```
//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
  }
```

115

115



*WHICH CODE HAS **INCORRECT SYNTAX**:*

```
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" );
```

116

116

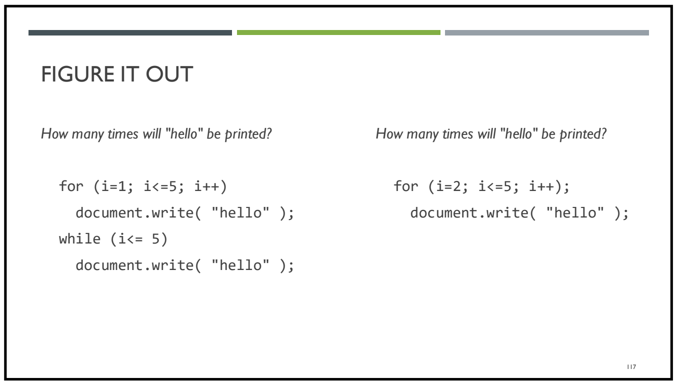


FIGURE IT OUT

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" );
```

117

117

ARRAYS IN JAVASCRIPT

- Arrays have a name that refers to the whole group
- Individual items (or *elements*) are identified by their ordinal position using an *index*
  - The first index is 0
- Array elements can be differing types
- The size of the array is *dynamic*
- Use the *Array* object to create an array

```
things = new Array();
```

118

A CONCEPTUAL VIEW …

- numbers = new Array(1,2,3,4,5);
- numbers = [1,2,3,4,5];

1
2
3
4
5

- *Create an array with 5 elements*
  - var numbers = new Array(5);

- document.write (numbers[1]);
  - Displays the 2nd element in the array – in this case "2"

119

THE ARRAY OBJECT

Create an empty array	var array = new Array();
Add elements to the end	array.push(1,2,3);
Change the element in the 3rd position	array[2]=5;
Add an element in the last position	array[3]=4;
Sort an array	array.sort()

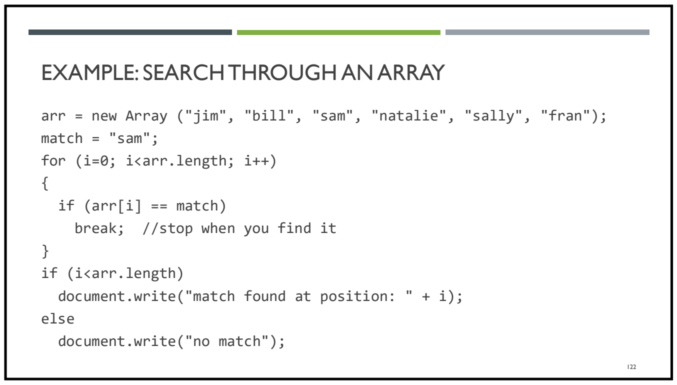
120

LOOPS AND ARRAYS

- A loop counter can serve as the index for an array.
- This makes it easy to *iterate* through every element in the array
  - To display the array
  - To do something to every element
  - To give a value to every element

```
//display an array
numbers = new Array(2,4,6,8,10);
for (i=0; i<numbers.length;i++)
  document.write(numbers[i]);
```

121



EXAMPLE: SEARCH THROUGH AN ARRAY

```
arr = new Array ("jim", "bill", "sam", "natalie", "sally", "fran");
match = "sam";
for (i=0; i<arr.length; i++)
{
  if (arr[i] == match)
    break;  //stop when you find it
}
if (i<arr.length)
  document.write("match found at position: " + i);
else
  document.write("no match");
```

122

122