1 **Statements (Conditions & Loops)**
JavaScript

2 **Conditional Statements (Page 1)**
- Provides *alternative* execution *paths*
- One path or another is chosen when the program executes depending upon a Boolean test result
  - Either True or False—the *only two* truth values
- Both selection (decision) and iteration (repetition/loop) structures use conditional statements

3 **The Relation Condition (Page 1)**
- A *booleanExpression* that compares two factors for conditions that are:
  - Equal to (or not equal to)
  - Greater than (or equal to)
  - Less than (or equal to)

4 **The Relation Condition (Page 2)**
- Format:
  factor1 relationalOperator factor2
  - At least one factor *must be a variable*—both may be
- Examples:
  salary >= 150
  hours == noon

5 **Relational Operators**
- Also sometimes called conditional operators:

==	Is equal to
<	Is less than
>	Is greater than
<=	Is less than or equal (identical) to
>=	Is greater than or equal (identical) to
!=	Is *not* equal to

6 **Single-Sided if Statement (Page 1)**
- Statement(s) executed *only if* the test is true
- If the condition is false, statement(s) in body of structure are *skipped*
- Format:

```
if (booleanExpression)
{
   statement(s) to be executed
   when the test succeeds go here;
}
```

7 **Single-Sided if Statement (Page 2)**
1. The keyword if
2. The *booleanExpression* inside (parentheses)
3. Statement(s) to be executed if *booleanExpression* is *true* in {braces}

8 **Flowchart for Single-Sided if Statement**

9 **Try It Out**
- conditionals1.htm

10 **Single-Sided if Statement (Page 3)**
- Example:

```
if (salary >= 150)
```

```
{
   richPoor = "rich";
}
```

- Semi-colon (;) placement—after every statement in the body of the structure (*not* after the condition)

11 **Single-Sided if Statement (Page 4)**

- Alterate if only one statement to be executed—
  - Alternate example 1:

```
if (salary >= 150)
   richPoor = "rich";
```

  - Alternate example 2:

```
if (salary >= 150) richPoor = "rich";
```

12 **Try It Out**

- conditionals2.htm

13 **Double-Sided if Statement (Page 1)**

- *One* (*set* of) statement(s) executed if the test evaluates as true
- *Another* (*set* of) statement(s) executed if the test is false

14 **Double-Sided if Statement (Page 2)**

- Format:

```
if (booleanExpression)
{
   statement(s) to be executed
   when the test succeeds go here;
}
else
{
   statement(s) to be executed
   when the test fails go here;
}
```

15 **Double-Sided if Statement (Page 3)**

1. The keyword if
2. The booleanExpression inside (parentheses)
3. Statement(s) to be executed if booleanExpression is *true* in the first set of {braces}
4. The keyword else
5. The statement(s) to be executed if the test *fails* in the second set of {braces}

16 **Flowchart for Double-Sided if Statement**

17 **Double-Sided if Statement (Page 4)**

- Example:

```
if (hoursWorked > 40)
{
   grossPay = 40 * payRate + (hoursWorked - 40) * payRate * 1.5;
}
else // hoursWorked <= 40
{
   grossPay = hoursWorked * payRate;
}
```

18 **Double-Sided if Statement (Page 5)**

- Example (alternate version):

```
if (hoursWorked > 40)
    grossPay = 40 * payRate + (hoursWorked - 40) * payRate * 1.5;
else
    grossPay = hoursWorked * payRate;
```

- Valid useage without {braces} because there is only one statement for each truth value …
  - Statement if true goes after the *condition*
  - Statement if false goes after the keyword else

19 **Double-Sided if Statement (Page 6)**

- Example (another alternate version):

```
if (hoursWorked > 40) grossPay = 40 * payRate + (hoursWorked - 40) * payRate *
                    1.5;
else grossPay = hoursWorked * payRate
```

20 **Try It Out**

- conditionals3htm

21 **Code Blocks (Page 1)**

- A set of statements inside {braces}
- *More than one* statement may executed within either true or false test block
- Format:

```
if (booleanExpression)
{
    statement1;
    statement2;
   [statement3; … ]
}
```

22 **Code Blocks (Page 2)**

- Example:

```
if (mouseX > 100)
{
    fill(255);
    background(0);
}
```

23 **The else if Structure**

- The if statement only can test for *two* truth values in an individual statement
- An option available for this purpose is the Linear Nested if …
  - It can test for *more than two* values (or *ranges* of values) of same variable within the structure
- Sometimes known as Case Selection

24 **Format of Linear Nested if**

```
if (booleanExpression1)
{
    statement(s) to be executed
    when test 1 succeeds go here;
}
else if (booleanExpression2)
{
    statement(s) to be executed
    when test 2 succeeds go here;
}
```

[else if …]

[else
{
statement(s) to be executed
when all tests fail go here;
}]

25 **Example of Linear Nested if**

```
if (now.getHours() < 12)
{
   document.write("<p>Good morning!</p>");
}
else if (now.getHours() < 18)
{
   document.write("<p>Good afternoon!</p>");
}
else // if (now.getHours() >= 18)
{
   document.write("<p>Good evening!</p>");
}
```

26 **Flowchart for Linear Nested if**

27 **Try It Out**

- conditionals4.htm

28 **The switch Statement (Page 1)**

- Controls program flow by executing a specific set of statements dependent on an expression value
- Compares expression value to value contained within a case label
- The case label
  - Represents a specific value
  - Contains one or more statements that execute if the case label value matches switch statement's expression value

29 **The switch Statement (Page 2)**

- default label
  - Executes when the value returned by the switch statement expression does not match a case label
- When a switch statement executes:
  - Value returned by the expression is compared to each case label in the order in which it is encountered
- break statement
  - Ends execution of a switch statement
  - Should be final statement after each case label

30 **The switch Statement (Page 3)**

- Format:

```
switch (expression)
{
   case label:
      statements;
      break;
   case label:
```

```
    statements;
    break;
  ...
  default:
    statements;
    break;
}
```

31 **The switch Statement (Page 3)**

- Example:

```
switch (secretNumber)
{
   case 1:
      document.write("<p>Too low!</p>");
      break;
   case 3:
      document.write("<p>You guessed the secret number!</p>");
      break;
   case 4:
      document.write("<p>Too high!</p>");
      break;
   default:
      document.write("<p>You did not enter a number between 1 and 5.</p>");
      break;
}
```

32 **Try It Out**

- conditionals5.htm

33 **Combined if Statements (Page 1)**

- A combined (also called compound) if statement contains …
  - Multiple Boolean expressions (*more* than one)
  - Connected by the logical operators …
    - && (meaning AND)
    - || (meaning OR)

34 **Combined if Statements (Page 2)**

- Formats:

```
if (booleanExpression1 && booleanExpression2)
                — or—
if (booleanExpression3 || booleanExpression4)
```

- Examples:

```
if (age >= 0 && age <= 10)
if (age <= 10 || age >= 80)
```

35 **Combined if Statements (Page 3)**

- With the && (AND) operator *all* conditions must be true
- With the !! (OR) operator *any one* condition must be true

36 **The ! (NOT) Logical Operator**

- Used to negate a Boolean expression so that the *inverse* (opposite) of the condition is true
- Format:

```
! booleanExpression
```

- The *booleanExpression* could be a Boolean variable, Boolean function or a relation condition

- Example:
  if (! (age >= 0 && age <= 10))
- Is equivalent to:
  if (age < 0 || age > 10)

37 **Try It Out**
- conditionals6.htm

38 **Conditional (Ternary) Operator (Page 1)**
- The conditional operator returns *one of two values* based on a Boolean condition
- The value returned if true follows the question mark (?) operator
- The value returned if false follows the colon (:) operator

39 **Conditional (Ternary) Operator (Page 2)**
- Format:
  (*booleanExpression*) ? *valueIfTrue* : *valueIfFalse*
  - *booleanExpession* is like an if condition
  - *valueIfTrue* is returned if the *booleanExpression* is true
  - *valueIfFalse* is returned if the *booleanExpression* is false

40 **Conditional (Ternary) Operator (Page 3)**
- Example:
  var eligible = (age < 18) ? "Too young" : "Old enough";
- Is equivalent to:

```
var eligible;
if (age < 18)
    eligible = "Too young";
else
    eligible = "Old enough";
```

41 **Try It Out**
- conditionals7.htm

42 **HTML Tables**
- Tables are a systematic arrangement of data in rows and columns similar to a spreadsheet (e.g. Microsoft Excel):
  - For displaying text, images, links, forms and form fields, other tables, etc.
- The basic unit of every table is the cell which is the intersection of a row and column and is the smallest component of a table

43 **Creating a Table in a Web Document**
- A table is a block inserted within the tags <table> … </table>
- Rows are inserted within the table using the tags <tr> … </tr>
- Cells are inserted into the row's using the tags <td> … </td>.

44 **Creating a Table Example**

```
<table>
    <tr>
        <td>Some data</td>
        <td>More data</td>
        <td>End of line</td>
    </tr>
<tr>
        <td>Even more data</td>
        <td>And more data</td>
```

```
        <td>End of second line</td>
    </tr>
</table>
```

45 **Adding Table Headers (Page 1)**

- Table headers are titles displayed in bold and centered above each of the columns of the table
- There usually should be as many headings as there are columns in the table

46 **Adding Table Headers (Page 2)**

- To create headers over each column, the developer uses <th> tags within the first <tr> (row) of the table
  - Usually headers are placed in the first column of rows
  - Theoretically headers could be placed into any row

47 **Adding Table Headers (Page 3)**

- Format:
  - <th>*content*</th>
- Example:
  -
```
<tr>
    <th>Monday</th>
    <th>Tuesday</th>
    <th>Wednesday</th>
    <th>Thursday</th>
    <th>Friday</th>
    <th>Week's Total</th>
</tr>
```

48 **Adding Table Borders (Page 1)**

- In a table the border is/are the lines that are displayed around the table and around all cells
- The border attribute sets the width of the border value as a valid CSS unit of measure

49 **Adding Table Borders (Page 2)**

- A value of zero (0) will turn off all borders around and within the table, the default in a Web table
- The border attribute for the table does not control the width of borders around cells, but must be set on for cell borders to display

50 **Adding Table Borders (Page 3)**

- Format:
  - <table border="*value*">
- Example:
  -
```
<table  border="5">
    <tr>
        <td>Some data</td>
        <td>More data</td>
        <td>End of line</td>
    </tr>
</table>
```

51 **Try It Out**

- loops3a.htm
- loops3b.htm

52 **Iteration (Loop) Statements**

- Provides *repeated* execution of a block of program statements
- The loop continues:
  - While a *condition* is met (sentinel-controlled looping) or …
  - A specified number of times (counter-controlled looping)
- Also called repetition or do while structure
  - *Meaning* do the loop while condition is True

53 **The while Loop (Page 1)**

- Continues to repeat a loop as long as a *controlling condition* is true (pre-test)
  - Sentinel-controlled looping
- The variable controlling the condition must be updated by logic within the loop
  - Exact number of loops is *usually unknown*

54 **The while Loop (Page 2)**

- Format:

```
while (booleanExpression)
{
    Statement(s) to be repeated as long as the booleanExpression is true;
}
```

55 **The while Loop (Page 3)**

- Example:

```
var balance = 250000;
var rate = .08;

while (balance < 1000000)
{
   balance *= (1 + rate);
   year++;
}
```

56 **The while Loop**

57 **Try It Out**

- loops1.htm

58 **The do while Loop (Page 1)**

- Continues to repeat a loop as long as a *controlling condition* is true
- Performs the test at the *conclusion* of the execution of each loop (post-test)
- Format:

```
do
{
    Statement(s) to be repeated as long as the booleanExpression is true;
}
while (booleanExpression);
```

59 **The do while Loop (Page 2)**

- Example:

```
var answer;
do
{
   …
   answer = prompt( "Another loop? (y/n)" );
}  while (answer = "y");
```

- A do while loop always executes *at least once*

60 **The do while Loop**

61 **Try It Out**
- loops2.htm

62 **The for Loop (Page 1)**
- Implements a loop by *counting* a specific number of iterations (repetitions)
  - Counter-controlled looping
- Appropriate when exact number of loop repetitions *is known*
- Format:

```
for (initialize; booleanExpression; increment)
{
    Statement(s) to be repeated;
}
```

63 **The for Loop (Page 2)**
- Example (three expressions in the parentheses):

```
for (ctr = 0; ctr <= 9; ctr++)
```

- The *initialize* component (ctr = 0)
  - Value assigned to a *counter* variable when the loop is *first encountered* in the program
- The *booleanExpression* component (ctr <= 9)
  - *Relation condition* which is tested to determine if the loop should be entered again
- The *increment* component (ctr++)
  - Indicates by what value the counter *changes* at the beginning of *each subsequent loop*

64 **The for Loop (Page 3)**
- Example:

```
var fahrenheit = [212, 72, 32, 0, -444];
var index, centigrade;

for (index = 0; index < 4; index++)
{
    centigrade = 5 / 9 * (fahrenheit[index] – 32);
    document.write(centigrade);
}
```

65 **Try It Out**
- loops3.htm

66 **The for…in Statement (Page 1)**
- Enables looping through an array without knowing exact number of elements
- During each loop execution the index for the array element is assigned to the for…in variable
- Format:

```
for (index in arrayName)
{
    statement(s)
}
```

67 **The for…in Statement (Page 2)**
- Example:

```
var fahrenheit = [212, 72, 32, 0, -444];
var index, centigrade;

for (index in fahrenheit)
```

```
    {
        centigrade = 5 / 9 * (fahrenheit[index] – 32);
        document.write(centigrade);
    }
```

68 **Try It Out**

- loops4.htm