

Computer Science Principles

Web Programming

JavaScript Programming Essentials

CHAPTER 6: CONDITIONALS AND LOOPS

DR. ERIC CHOU

IEEE SENIOR MEMBER



Overview

LECTURE 1



Scope of the Lesson

- Conditional Statements
 - The if statement
 - The else statement
 - The switch statement
 - The break Keyword
- Different Kind of Loops
 - JavaScript for loop
 - JavaScript for/in loop
 - JavaScript while loop
 - JavaScript do/while loop



Objectives

- By the end of the lesson, you will be familiar and know how the website works using JavaScripts.
 - Discuss the introduction to JavaScript and using conditional statements.
 - Understand the coding syntax using the break keywords.
 - Explain thoroughly the coding styles of different kinds of loops.



Conditional Statements

LECTURE 1



Conditional Statements

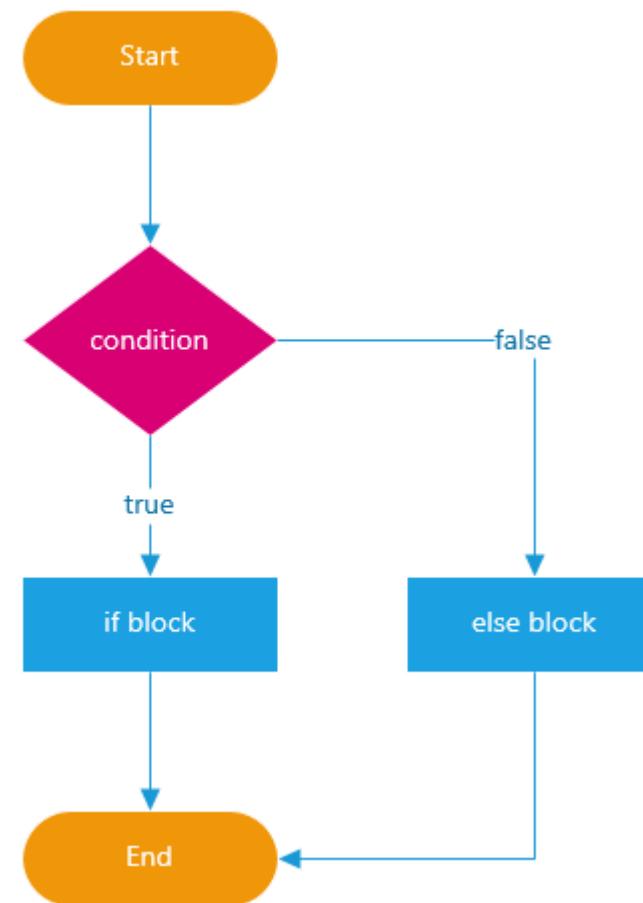
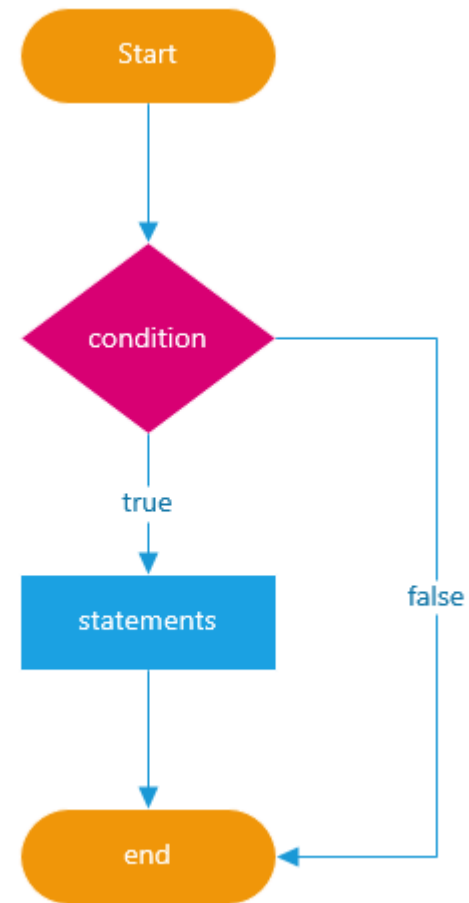
Conditional statements are used to perform different actions based on different conditions.

- Very often when we write code, we wanted to perform different actions for different decisions.
- We can use conditional statements in our code to do this.



Conditional Statements

- 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.





The if Statement

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

- **Syntax:**

```
if (condition) {  
    block of code to be executed if  
    the condition is true  
}
```

- Note that **if** is in lower case. Uppercase letters will generate a JavaScript error.



The if Statement

- Example:

Make a “Good day” greeting if the hour is less than 18:00:

```
if (hour < 18) {  
    greeting = "Good day";  
}
```

- The result of greeting will be:

Good day



The **else** Statement

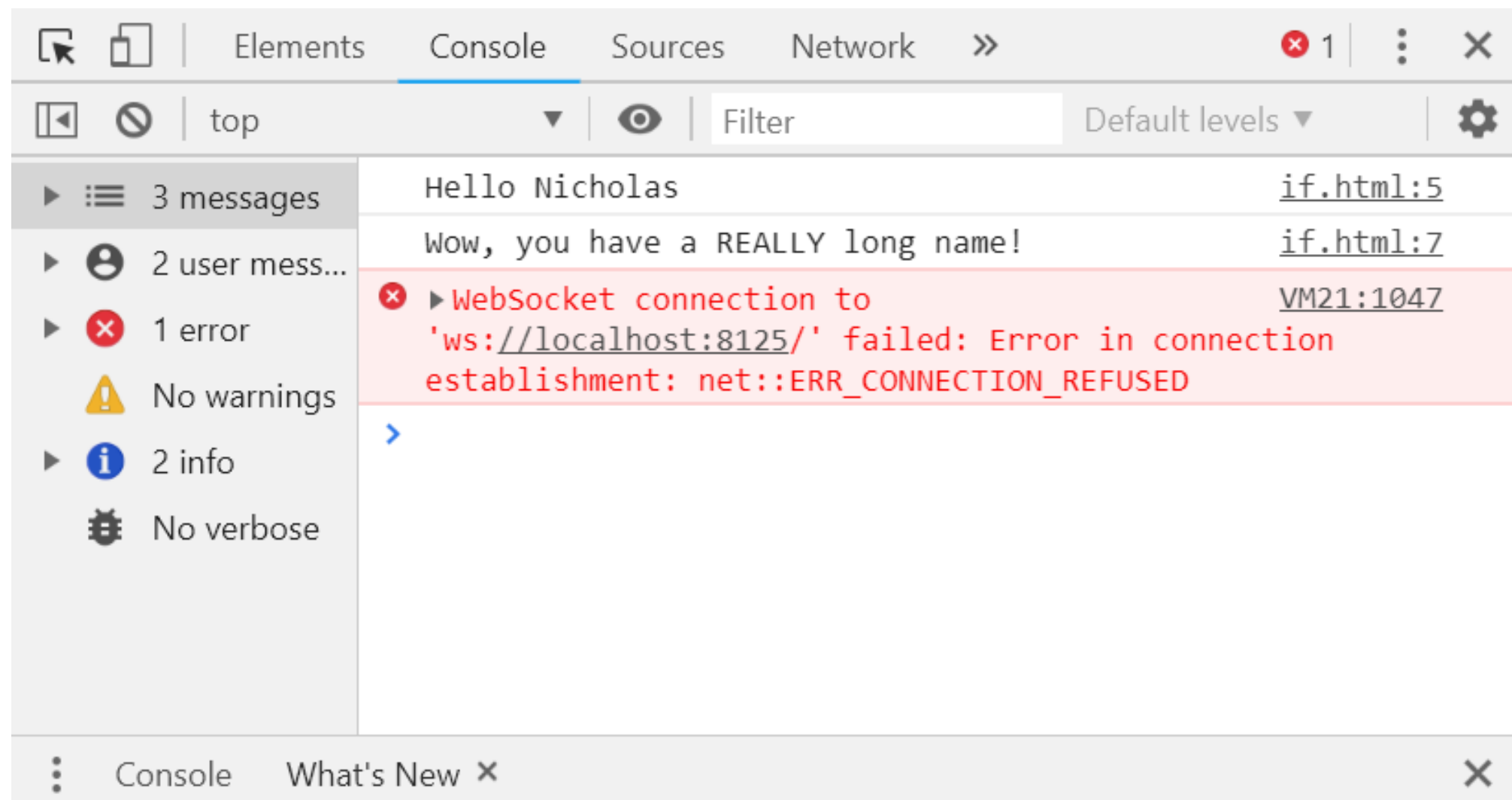
- Use **else** to specify a block of code to be executed, if the same condition is false.

```
if (condition) {  
    block of code to be executed if  
    the condition is true  
}  
else {  
    block of code to be executed if  
    the condition is false  
}
```



Demo Program: if.html

```
1 ▼ <html>
2 ▼   <body>
3 ▼   <script>
4       var name = "Nicholas";
5       console.log("Hello " + name);
6 ▼       if (name.length > 7) {
7           console.log("Wow, you have a REALLY long name!");
8       }
9   </script>
10  </body>
11 </html>
```





The `else` Statement

- **Example:**

- If the hour is less than 18, create a "Good day" greeting, otherwise "Good evening":

```
if (hour < 18) {  
  greeting = "Good day";  
} else {  
  greeting = "Good evening";  
}
```

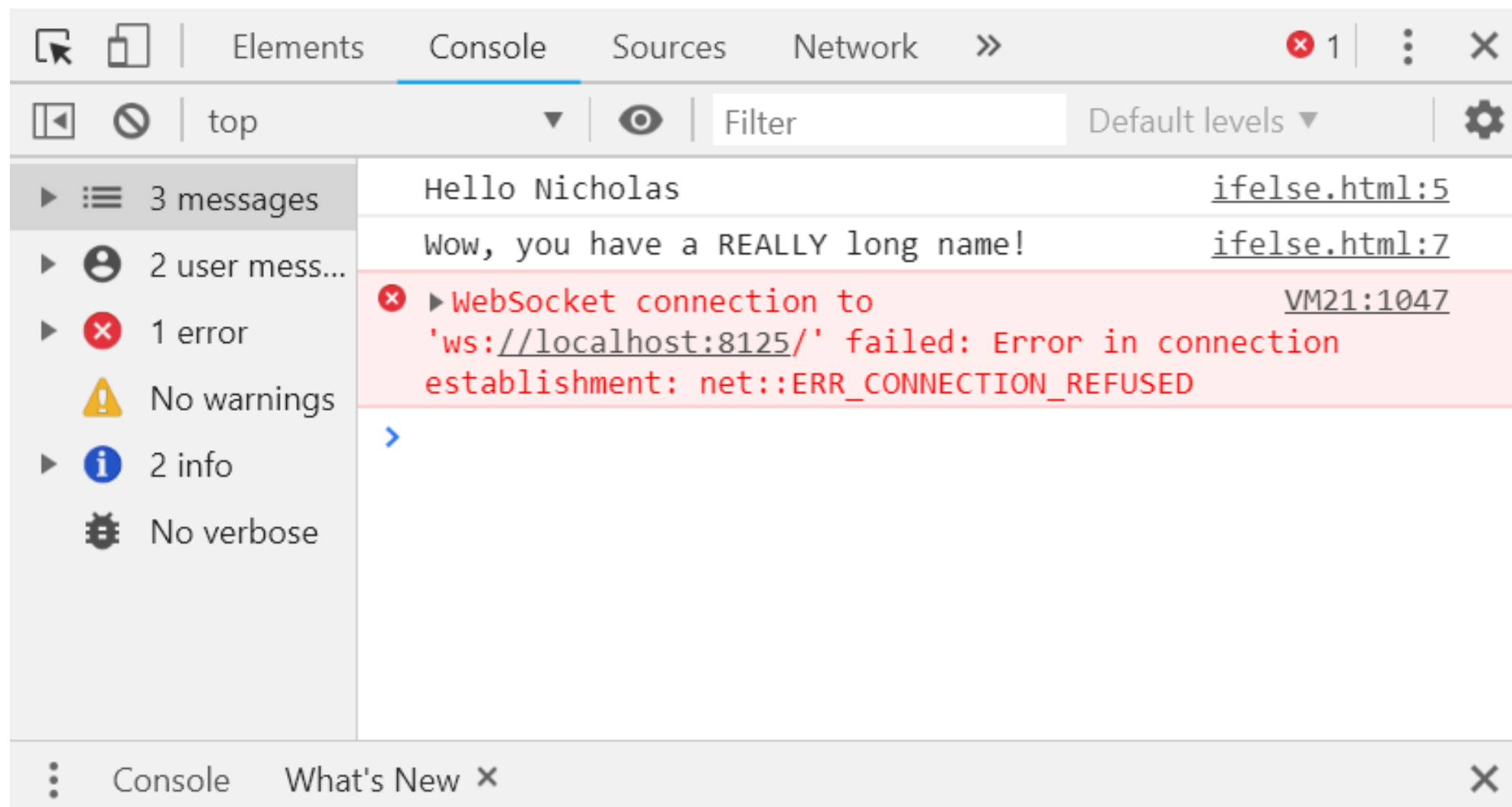
- The result of greeting will be:

Good day



Demo Program: ifelse.html

```
1 ▼ <html>
2 ▼   <body>
3 ▼   <script>
4       var name = "Nicholas";
5       console.log("Hello " + name);
6 ▼       if (name.length > 7) {
7           console.log("Wow, you have a REALLY long name!");
8       }
9 ▼       else {
10          console.log("Your name isn't very long.");
11      }
12   </script>
13   </body>
14 </html>
```





The **else if** Statement

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

```
if (condition1) {  
    block of code to be executed if  
    condition1 is true  
} else if (condition2) {  
    block of code to be executed if  
    the condition1 is false and condition2 is true  
} else {  
    block of code to be executed if  
    the condition1 is false and condition2 is false  
}
```



The **else if** Statement

- **Example:**

- If time is less than 10:00, create a “Good morning” greeting, if not, but time is less than 20:00, create a “Good day” greeting, otherwise a "Good evening":

```
if (time < 10) {  
  greeting = "Good morning";  
} else if (time < 20) {  
  greeting = "Good day";  
} else {  
  greeting = "Good evening";  
}
```



Demo Program: ifelseChained.html

- Nested if-else-if-else statements.
- Maybe replaced by switch-statement in some situations.

```
1 ▼ <html>
2 ▼   <body>
3 ▼   <script>
4       var lemonChicken = false;
5       var beefWithBlackBean = true;
6       var sweetAndSourPork = true;
7 ▼     if (lemonChicken) {
8         console.log("Great! I'm having lemon chicken!");
9 ▼     } else if (beefWithBlackBean) {
10         console.log("I'm having the beef.");
11 ▼     } else if (sweetAndSourPork) {
12         console.log("OK, I'll have the pork.");
13 ▼     } else {
14         console.log("Well, I guess I'll have rice then.");
15     }
16   </script>
17 </body>
18 </html>
```

127.0.0.1:57436/pass/ifelseChained

127.0.0.1:57436/pass/ifelseChained.html

Apps

Elements

Console

Sources

Network

Performance

1

top

Filter

Default levels

2 messages

1 user mess...

1 error

No warnings

1 info

No verbose

I'm having the beef.

ifelseChained.html:10

Failed to load resource: the server responded with a status of 404 (Not Found)

:57436/favicon.ico:1

>

Console

What's New

Highlights from the Chrome 74 update



Switch

LECTURE 1



The **switch** Statement

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

- **Syntax:**

```
switch(expression) {  
    case n:    /* code block */ break;  
    case n:    /* code block */ break;  
    default:  /* default code block */  
}
```



The **switch** Statement

- How switch works?
- The switch expression is evaluated once.
- The value of the expression is compared with the value of each case.
- If there is a match, the associated block of code is executed.



The **switch** Statement

- **Example:**
 - The `getDay()` method returns the weekday as a number between 0 and 6. (Sunday=0, Monday=1, Tuesday=2 ..)
 - Use the weekday number to calculate weekday name.



Switch

Demo Program: switch.html

- 6-way switch.
- String as switch variable
- Output to document.

```
1 ▼ <html>
2 ▼   <body>
3 ▼     <script type = "text/javascript">
4       var grade = 'A';
5       document.write("Entering switch block<br />");
6 ▼     switch (grade) {
7       case 'A': document.write("Good job<br />");      break;
8       case 'B': document.write("Pretty good<br />");  break;
9       case 'C': document.write("Passed<br />");       break;
10      case 'D': document.write("Not so good<br />");   break;
11      case 'F': document.write("Failed<br />");       break;
12      default:  document.write("Unknown grade<br />")
13    }
14    document.write("Exiting switch block");
15  </script>
16  <p>Set the variable to different value and then try...</p>
17 </body>
18 </html>
```





Conditional Operator

LECTURE 1



JavaScript if else shortcut: conditional operator

- JavaScript provides a conditional operator that can be used as a shortcut of the if statement. The following illustrates the syntax of the conditional operator.

condition ? expression_1 : expression_2

- Like the if statement, the condition is an expression that evaluates to true or false. If the condition evaluates to true, the operator returns the value of the expression_1; otherwise, it returns the value of the expression_2.



JavaScript if else shortcut: conditional operator

- For example, to display a different label for the login button based on the value of the `isLoggedIn` variable, you could use the conditional operator as follows:

```
isLoggedIn ? "Logout" : "Login";
```

- You can also assign a variable depending on the result of the ternary operator.

```
// only register if the age is greater than 18  
var allowRegister = age > 18 ? true : false;
```



JavaScript if else shortcut: conditional operator

- If you want to do more than a single operation per case, you need to separate operation using a comma (,) as the following example:

```
age > 18 ? (  
    alert("OK, you can register."),  
    redirectTo("register.html");  
) : (  
    stop = true,  
    alert("Sorry, you are too young!")  
);
```

- In this tutorial, you have learned how to use the JavaScript if else statement to execute a statement when a condition evaluates to true and executes another statement when the condition evaluates to false.



Loops

LECTURE 1



Different Kinds of Loops

- 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 specified condition is true.



JavaScript **for** loop

- 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. Often this is the case when working with arrays:

```
text += cars[0] + "<br>";  
text += cars[1] + "<br>";  
text += cars[2] + "<br>";  
text += cars[3] + "<br>";  
text += cars[4] + "<br>";  
text += cars[5] + "<br>";
```



JavaScript **for** loop

- 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 (statement 1; statement 2; statement 3) {  
    code block to be executed  
}
```



JavaScript for loop

- Often this is the case when working with arrays:
- We can write:

```
text += cars[0] + "<br>";  
text += cars[1] + "<br>";  
text += cars[2] + "<br>";  
text += cars[3] + "<br>";  
text += cars[4] + "<br>";  
text += cars[5] + "<br>";  
for (i = 0; i < cars.length; i++) {  
text += cars[i] + "<br>";
```



JavaScript **for** loop

- Statement1 is executed before the loop (the code block) starts.
- Statement2 defines the condition running the loop (the code block)
- Statement3 is executed each time after the loop (the code block) has been executed.
- **Example:**

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



JavaScript **for/in** loop (for-each-attribute)

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

```
var person = {fname:"John", lname:"Doe", age:25};  
var text = "";  
var x;  
for (x in person) {  
    text += person[x];  
}
```



JavaScript **while** loop

- The while loop loops through a block of code as long as a specified condition is true.
- **Syntax:**

```
while (condition) {  
  code block to be executed  
}
```




JavaScript **while** loop

- **Example:**

- The code in the loop will run over and over again, as long as variable (i) is less than 10.

```
while (i < 10) {  
    text += "The number is " + i;  
    i++;  
}
```



JavaScript do/while loop

- 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.

- **Syntax:**

```
do {  
    code block to be executed  
}  
while (condition);
```



JavaScript do/while loop

- Example:**

- The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

```
do {  
    text += "The number is " + i;  
    i++;  
}  
while (i < 10);
```



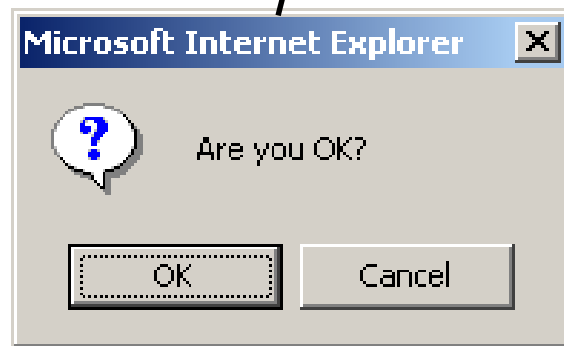
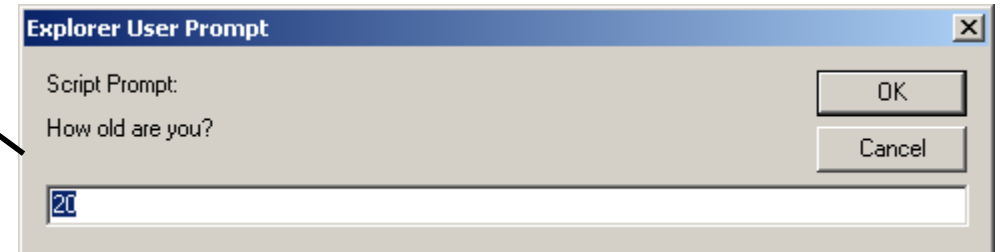
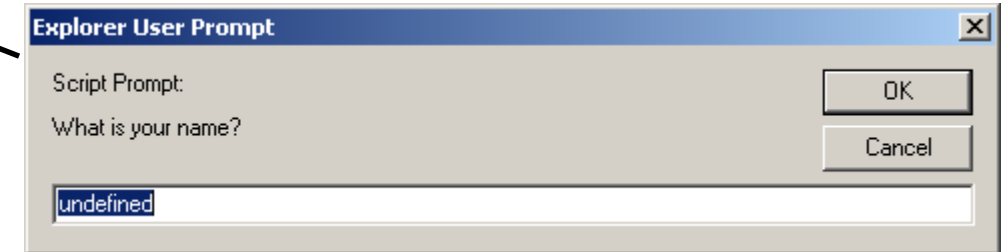
Input/Output with Browsers

LECTURE 1



alert(), confirm(), and prompt()

```
<script type="text/javascript">  
alert("This is an Alert method");  
confirm("Are you OK?");  
prompt("What is your name?");  
prompt("How old are you?", "20");  
</script>
```





alert() and confirm()

```
alert("Text to be displayed");
```

- Display a message in a dialog box.
- The dialog box will block the browser.

```
var answer = confirm("Are you sure?");
```

- Display a message in a dialog box with two buttons: "OK" or "Cancel".
- `confirm()` returns **true** if the user click "OK". Otherwise it returns **false**.



prompt ()

```
prompt("What is your student id number?");  
prompt("What is your name?", "No name");
```

- Display a message and allow the user to enter a value
- The second argument is the "default value" to be displayed in the input textfield.
- Without the default value, "undefined" is shown in the input textfield.
- If the user click the "OK" button, **prompt ()** returns the value in the input textfield as a string.
- If the user click the "Cancel" button, **prompt ()** returns null.



Break Levels

LECTURE 1

break



The **break** Keyword

- When the JavaScript code interpreter 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.
 - Note: When a match is found, and the job is done, it is time for a break. There is no need for more testing.



Label statement

- Before discussing the break statement, let's talk about the label statement first.
- In JavaScript, you can label a statement for later use. The following illustrates the syntax of the label statement.

```
1 label: statement;
```

- The label can be any valid identifier.
- The following example labels the loop using the outer label.

```
1 outer: for (var i = 0; i < 5; i++) {  
2     console.log(i);  
3 }
```

- You can reference to the label by using the break or continue statement. Typically, you use the label with nested loop such as for, do-while, and while loop.



JavaScript break statement

Demo Program: [break.html](#)

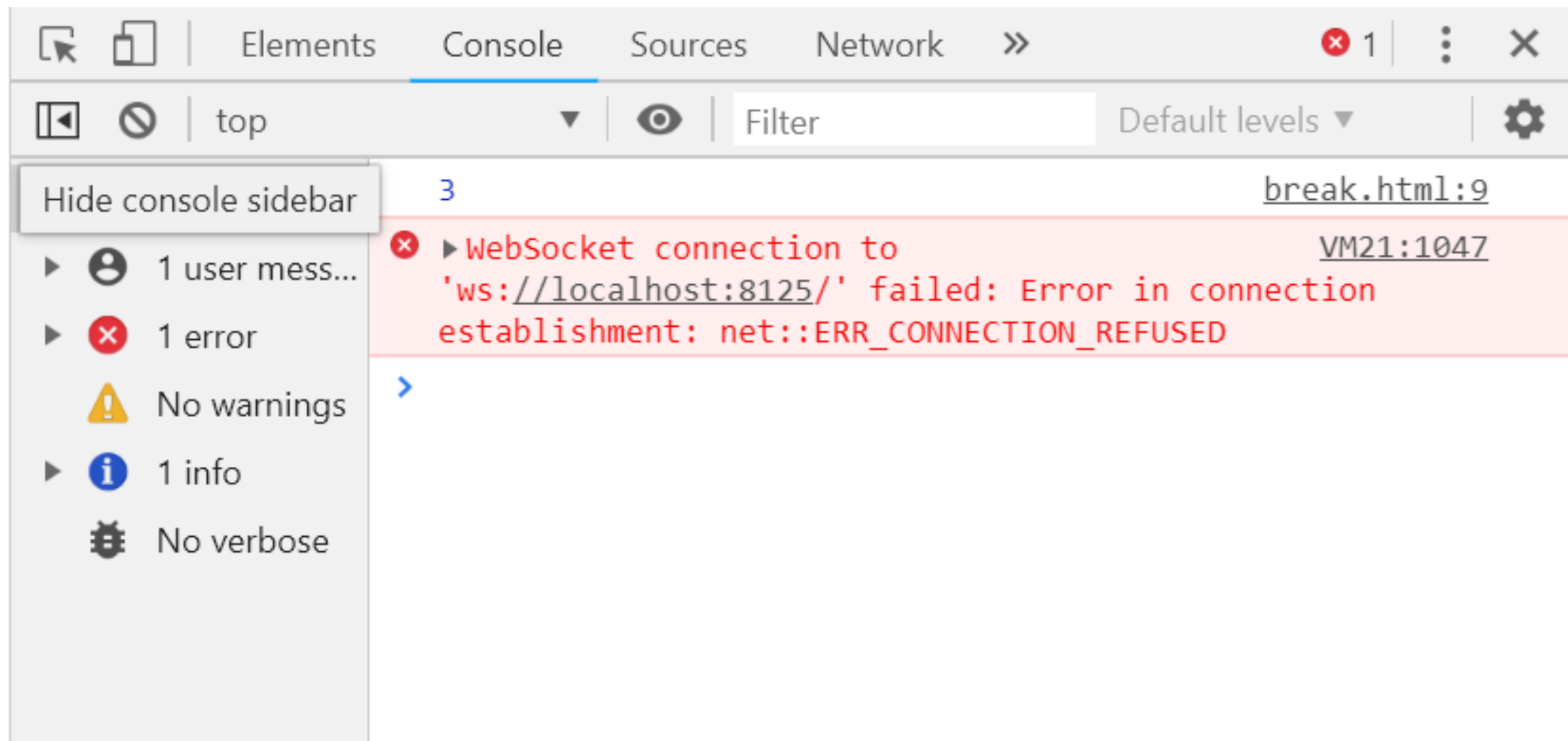
- The break statement gives you a fine-grained control over the execution of the code in a loop. The break statement terminates the loop immediately and passes control over the next statement after the loop.
- Here's an example.

```
1 for (var i = 1; i < 10; i++) {  
2     if (i % 3 == 0) {  
3         break;  
4     }  
5 }  
6 console.log(i); // 3
```



JavaScript break statement

- In this example, the `for` loop increments the variable `i` from 1 to 10. In the body of the loop, the if statement checks if `i` is evenly divisible by 3. If so, the break statement is executed and the loop is terminated.
- The control is passed to the next statement outside the loop that outputs the variable `i` to the web console.
- Besides controlling the loop, you also use the break statement to terminate a case branch in the switch block.





Using **break** statement to exit nested loop

- As mentioned earlier, you use the break statement to terminate a label statement and transfer control to the next statement following the terminated statement. The syntax is as follows:

`break label;`





Using **break** statement to exit nested loop


Demo Program: [break2.html](#)

- The `break` statement is typically used to exit the nested loop. See the following example.


```
1 var iterations = 0;
2 top: for (var i = 0; i < 5; i++) {
3     for (var j = 0; j < 5; j++) {
4         iterations++;
5         if (i === 2 && j === 2) {
6             break top;
7         }
8     }
9 }
10 console.log(iterations); // 13
```


Elements




Console




Sources




Network




>>





1




:



X




top



Filter

Default levels





2 messages



1 user mess...



1 error



No warnings



1 info



No verbose

13

break2.html:13



▶ WebSocket connection to VM21:1047
'ws://localhost:8125/' failed: Error in connection
establishment: net::ERR_CONNECTION_REFUSED

>

|

continue



JavaScript **continue**: Skips the Current Iteration of a Loop

- **Summary:** in this tutorial, you will learn how to use the JavaScript continue statement to skip the current iteration of a loop.
- The **continue** statement skips the current iteration of a loop and goes to the next one. Because of this, the continue statement must appear in the body of a loop or you will get an error.
- Similar to the **break** statement, the continue statement has two forms: labeled and unlabeled. For more information on the label statement, see the break statement tutorial.



Using unlabeled JavaScript **continue** statement

- The unlabeled **continue** statement skips the current iteration of a **for**, **do-while**, or **while loop**. The **continue** statement skips the rest of the code to the end of the innermost body of a loop and evaluates the expression that controls the loop.



Using unlabeled JavaScript **continue** statement

- In a for loop, the continue skips all the statements underneath it and pass the execution of the code to the update expression, in this case, it is **i++**;

```
1  for (var i = 0; i < count; i++) {  
2      if (condition)  
3          continue; // Jumps to expression: i++  
4      // more statement here  
5  }
```



Using unlabeled JavaScript `continue` statement

- In a while or do-while loop, it jumps back to the expression that controls the loop.

```
1 while (expression) // continue jumps here
2 {
3     if (condition) {
4         continue; // Jumps to expression
5     }
6     // more statements here
7     // ...
8 }
```

```
1 do{
2     if (condition) {
3         continue; // Jumps to expression
4     }
5     // more statements here
6     // ...
7 }while(expression); // continue jumps here
```



Using unlabeled JavaScript **continue** statement

Demo Program: continue.html (3)

```
1 var s = 'This is a JavaScript continue statement demo.';
2 var counter = 0;
3 for (var i = 0; i < s.length; i++) {
4     if (s.charAt(i) != 's') {
5         continue;
6     }
7     //
8     counter++;
9 }
10 console.log('The number of s found in the string is ' + counter);
```



Using JavaScript `continue` with a label

The `continue` statement can include an optional label as follows:

```
1 continue label;
```

The `label` can be any valid identifier.

See the following example.

```
1 // continue with a label
2 outer: for (var i = 1; i <= 3; i++) {
3     for (var j = 1; j <= 3; j++) {
4         if ((i == 2) && (j == 2)) {
5             console.log('continue to outer');
6             continue outer;
7         }
8         console.log("[i:" + i + ",j:" + j + "]");
9     }
10 }
```


pass in JavaScript



pass in JavaScript

Demo Program: pass.html

- Use single ; symbol as pass statement;

```
1 ▼ <html>
2 ▼   <body>
3 ▼   <script>
4     var i=0;
5     document.write("<h1>Demo of Pass: </h1>");
6     // single ; can be used for pass
7 ▼   if (i==0) {
8       ;
9   }
10    document.write("Passed...");
11  </script>
12  </body>
13  </html>|
```

return



Homework

LECTURE 1



Homework

- Download JS6HW1.pdf and work on it.
- There are 3 problems. animal.html, randomstring.html, and H4CK3R.html