Unit 3 Module 2: Intermediate JavaScript

Dylan Lane McDonald

CNM STEMulus Center
Web Development with PHP

August 25, 2014

Outline

Exception Handling

Definition

An **exception** *is an error condition in a program that precludes its normal execution. When an exception occurs, it is* **thrown**. *When a thrown exception is given back to the calling function, it is* **caught**.

Exception Handling

Definition

An **exception** *is an error condition in a program that precludes its normal execution. When an exception occurs, it is* **thrown**. *When a thrown exception is given back to the calling function, it is* **caught**.

Exceptions contain data as to what caused the exception and why. This gives the developer enormous flexibility on how to react to the exception when it is caught.

Exception Handling

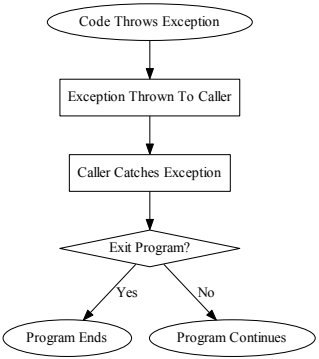

Figure 1: Exception Handling Flow

Exception Handling

```
try {
    // code to try under ''normal'' cases
    throw("Exception has occurred");
} catch(exception) {
    // code that happens when exception is thrown
}
```

Listing 1: Exception Handling

Variable Scope

Functions work independently of each other. Any variables within functions only exist while the function is running and cease to exist when the function exits. This is known as **variable scope**.

Variable Scope

Functions work independently of each other. Any variables within functions only exist while the function is running and cease to exist when the function exits. This is known as **variable scope**.

Myth: A variable x in function A can be accessed as x from function B.

Variable Scope

Functions work independently of each other. Any variables within functions only exist while the function is running and cease to exist when the function exits. This is known as **variable scope**.

Myth: A variable x in function A can be accessed as x from function B.
Fact: Function A has no idea about function B's variables. The fact x exists in both functions is irrelevant. There is no way for either function to access the other function's x.

Variable Scope

Functions work independently of each other. Any variables within functions only exist while the function is running and cease to exist when the function exits. This is known as **variable scope**.

Myth: A variable x in function A can be accessed as x from function B.
Fact: Function A has no idea about function B's variables. The fact x exists in both functions is irrelevant. There is no way for either function to access the other function's x. **Unless...**

Global Variables

Definition

A **global variable** *is a variable that is declared outside a function and is accessible to* **all** *functions in the file. All system variables are global variables.*

Global Variables

Definition

A **global variable** *is a variable that is declared outside a function and is accessible to* **all** *functions in the file. All system variables are global variables.*

Global variables are useful for data that must be shared among functions, but introduce a a very high probability for bugs. Global variables should be used very conservatively and only when absolutely necessary. The preferred way to share data between functions is to have the functions call other functions.

Global Variables

```
// globals are all caps by convention
var GLOBAL_X;

function firstFunction() {
    var x;
}

function secondFunction() {
    var x;
}
```

Listing 2: Global Variables

Definition

An **array** *is a collection of variables. It is used as a container to hold like or related objects.*

Definition

An **array** *is a collection of variables. It is used as a container to hold like or related objects.*

Arrays are indexed from 0, not 1. So the first element occurring in array is element 0. Arrays in JavaScript are dynamically sized and can be added to as the program executes.

Arrays

JavaScript has several functions for working with arrays. Arrays are *indexed* by using the []s to refer to a single element or the array name to refer to the entire array.

```
cars = new Array("BMW", "Honda", "Toyota");
console.log(cars[0]); // BMW
cars[3] = "Mazda";    // adds another car
```

Listing 3: Creating & Accessing an Array

[1] The 0^{th} element is often called the "first" element.

Arrays

JavaScript has several functions for working with arrays. Arrays are *indexed* by using the []s to refer to a single element or the array name to refer to the entire array.

```
cars = new Array("BMW", "Honda", "Toyota");
console.log(cars[0]); // BMW
cars[3] = "Mazda";    // adds another car
```

Listing 4: Creating & Accessing an Array

The first line of code creates an array *en masse*. The second line of code prints the first[1] car to the console. The third line adds another car and makes it the $4^{\underline{\text{th}}}$ element of the array.

[1] The $0^{\underline{\text{th}}}$ element is often called the "first" element.

Loops

Definition

A **loop** *is a block that is intended to be repeated multiple times. A loop will run from a start condition until an ending condition.*

Loops

Definition

A **loop** *is a block that is intended to be repeated multiple times. A loop will run from a start condition until an ending condition.*

Features of loops:

- **Start Condition**: When the loop starts execution
- **End Condition**: When the loop ceases execution
- **Step Condition**: When the loop advances from one iteration to the next, how long it should advance
- **Continue**: Advance the loop directly to the next iteration
- **Break**: Halt the loop's execution early

do...while Loop

```
do {
    // code to loop
} while (endCondition);
```

Listing 5: do...while Loop

do...while Loop

```
do {
    // code to loop
} while (endCondition);
```

Listing 6: do...while Loop

A do...while loop automatically starts and has no start condition. The end condition is tested after each iteration is executed. For this reason, it is known as a *post-test loop*.

while Loop

```
while(endCondition) {
    // code to loop
}
```

Listing 7: while Loop

while Loop

```
while(endCondition) {
    // code to loop
}
```

Listing 8: while Loop

A while loop starts if the end condition is true and has no start condition. The end condition is tested before each iteration is executed. For this reason, it is known as a *pre-test loop*.

for Loop

```
for(startCondition; endCondition; stepCondition) {
    // code to loop
}
```

Listing 9: for Loop

for Loop

```
for(startCondition; endCondition; stepCondition) {
    // code to loop
}
```

Listing 10: for Loop

A for loop starts with a start condition and steps according to the step condition. The loop terminates at the specified end condition. The main use of a for loop is for iterating through arrays.

for Loop with an Array

```
cars = new Array("BMW", "Honda", "Toyota");
for(i = 0; i < cars.length; i++) {
    console.log("I drive a " + cars[i]);
}
```

Listing 11: for Loop with an Array

for Loop with an Array

```
cars = new Array("BMW", "Honda", "Toyota");
for(i = 0; i < cars.length; i++) {
    console.log("I drive a " + cars[i]);
}
```

Listing 12: for Loop with an Array

Note the use of `cars.length`. The length property is a system property of arrays and will always return the size of the array.