



WESTCLIFF
UNIVERSITY

Educate. Inspire. Empower.

Week 3: JavaScript Review



Agenda

- What are variables & how to create and use them
- JavaScript data types
- What are functions & how to create and use them
- What are operators and their meanings
- What is Decision Making and the use of if-else if-else and switch statements
- What are loops, its construct and use
- What are arrays, its construct and use
- The purpose of popup boxes, its construct and use
- Lab Assignment
- Homework

Variables



Variables

- **Variable** is piece of code in JavaScript that can store data so that the program can use it later. Think of variable as a storage container.
- The value or data stored in a variable is not always fixed. It can be changed.
- Some of the rules for creating/using **variables**:
 - Uses reserved keyword **var** to declare/create a new variable.
 - Must have a unique name.
 - When assigning a value/data to a variable, use the equal (=) operator when declaring a variable.
 - Can leave variable empty (no data assigned) when declaring.
 - Must be declared/exist before using in the program.



How to create Variables

- To create a variable is also known as declaring a variable.
- Use the keyword **var** which is a reserved word. Here's how:

var *variableName*

- this creates an empty variable named *variableName*

var *variableName* = data

- this creates a variable named *variableName* with an assigned value of data

- Examples:

var *name* = "John";

var *age* = 25;

JavaScript Data types



JavaScript Data types

- Datatypes in JavaScript describe the different types or kinds of data that the program can work with and store in variables.
- There are five basic, or primitive, types of data and they are:
 - Strings
 - Numbers
 - Booleans
 - Undefined
 - Null



JavaScript Data types

- Strings

- A collection of alphanumeric characters, enclosed in quotation marks.
- Example of alphanumerics: a person's name, street number, email, etc.
- Examples: "John" , "5023" , "johndoe@gmail.com"

- Numbers

- Digits including negatives and decimals. They can be used for mathematical calculations.
- Example of digits: a person's age, temperature, total sales, etc
- Examples: 34, -10, 188.95



JavaScript Data types

- Boolean

- Boolean is like a on/off switch.
- It has two values – true or false.

- Null

- Basically null means empty or no value.
- If a variable is assigned null ie: `var name = null`, that means the variable *name* contains a value but it's empty.

- Undefined

- Undefined happens when a program is requesting data from a variable that has not been assigned any value in it or the variable hasn't been declared yet.

Functions



How Functions are created

- The basic syntax:

```
function functionName() {  
    some lines of codes to be executed  
}
```

- Functions can also accept a parameter:

```
function functionName(parameter) {  
    some lines of codes to be executed  
}
```

- Or multiple parameters:

```
function functionName(parameter1, parameter2, parameter3) {  
    some lines of codes to be executed  
}
```



How Functions are created and used

Examples:

1. Create variable that contains the data:

```
//Declaring a variable  
var name = "John";
```

1. Creating a function that accepts a parameter:

```
//Defining a function  
function newStudent(name) {  
    document.write("New student " + name + " is added to the system.");  
}
```

1. Run the function:

```
//Run function  
newStudent(name);
```

Operators



What are operators and their meanings

Arithmetic Operators

+	addition
-	subtraction
*	multiplication
/	division
++	increment by 1
--	decrement by 1

Examples:

```
var a = 10;  
var b = 5;
```

```
a + b = 15  
a++ will give 11  
b- - will give 4
```



What are operators and their meanings

Comparison Operators

==	equal
!=	not equal
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

* Comparison operators check by comparing 2 data with an outcome of true or false

Examples:

```
var a = 10;  
var b = 5;  
var c = 5;
```

a == b	will give false
a != b	will give true
b > a	will give false
b <= c	will give true



What are operators and their meanings

Logical Operators

&& logical AND
|| logical OR
! logical NOT

* Logical operators check by comparing 2 pieces of data or the result of it with an outcome of true or false

Examples:

```
var a = true;  
var b = false;
```

a && b will give false
a || b will give true
!(b && a) will give true
!(a || b) will give false



What are operators and their meanings

Assignment Operators

=	assign
+=	add and assign
-=	subtract and assign
*=	multiply and assign
/=	divide and assign

Examples:

```
var a = 10;  
var b = 5;  
var result;
```

result = (a = b)	will give 5
result = (a += b)	will give 15
result = (a -= b)	will give 5
result = (a *= b)	will give 50
result = (a /= b)	will give 2



What are operators and their meanings

Conditional or Ternary Operator

`?:` use for testing if a condition is true, output X if true and output Y if false

Syntax:

(condition) `?` X `:` Y

Examples:

```
var a = 10;  
var b = 5;
```

`(a > b) ? true : false;` will give true

`(a < b) ? a : b;` will give 5

Decision Making



How Decision Making is used

- First, let's look at the basic syntax of a simple **if** statement:
if (condition) { action }
- An **if – else** statement:
if (condition) { action 1 } else { action 2 }
- An **if – else if – else** statement:
if (condition 1) { action 1 } else if (condition 2) { action 2 } else { action 3 }
- You have the option to create multiple **else if** conditions if required



How Decision Making is used

- A simple example of how if-else is use in JavaScript:

```
<h2>JavaScript Conditional Statement</h2>
<p>Example of if-else conditions</p>

<script>
//Declaring a variable
var name = "John";

//if else statement
if (name == "John") {
    document.write("Student name is " + name);
} else {
    document.write("It's a different student. His/her name is " + name);
}
</script>
```

- What it looks like when view on the browser:

JavaScript Conditional Statement

Example of if-else conditions

Student name is John



How Decision Making is used

- Next we will discuss the other conditional statement: **switch**.
- Like if - else statement, switch could also evaluate multiple conditions.
- First, let's look at the basic syntax of a simple **switch** statement:

```
switch(expression) {  
    case x:  
        // code block  
        break;  
    case y:  
        // code block  
        break;  
    default:  
        // code block  
}
```

expression - what are you evaluating
case x, case y – different conditions
default – same as *else* in if statement
break – exit if that condition is evaluate to true



How Decision Making is used

- A simple example of how `switch` is use in JavaScript:

```
<h2>JavaScript Conditional Statement</h2>
<p>Example of Switch statement</p>

<script>
//Declaring a variable
var name = "Hanna";

//switch statement
switch(name) {
  case 'John': // if (name == 'John')
    document.write("Student name is " + name);
    break;

  case 'Hanna': // if (name == 'Hanna')
    document.write("It's a different student. His/her name is " + name);
    break;

  default:
    document.write("No such student name exist");
}
</script>
```

What it looks like when view
on the browser:

JavaScript Conditional Statement

Example of Switch statement

It's a different student. His/her name is Hanna

Loops



How Loops are used

- Let's look at the basic syntax for first type of loop: **for** loop

```
for (initialization; condition; counter) {  
    // code block  
}
```

- Breakdown:
 - *initialization*: declare the counter variable and its starting value
 - *condition*: the expression that evaluates to true or false
 - *counter*: increment or decrement of the count
- Common use:
 - count through a finite number of steps or tasks



How Loops are used

- Be careful of using the incorrect operators in the condition:

```
<h2>JavaScript Loops</h2>
<p>Example of for Loop</p>
```

```
<script>
//For loop
for (var i = 0; i < 10; i++) {
    document.write(i + " ");
    console.log(i);
}
</script>
```

<=

if equal sign left out

What it looks like when view on the browser:

JavaScript Loops

Example of for Loop

0 1 2 3 4 5 6 7 8 9 10

10 is omitted



How Loops are used

- Let's look at the basic syntax of `while` loop

```
//While loop
while (condition) {
    //code block
}
```

- Explain:
 - `while` loop continues to run as long as the condition is *true*
 - loop ends if condition evaluates to *false*
- Common use:
 - to determine when a task(s) should continue or ends depending on the condition.



How Loops are used

- Example of **while** loop: say you want to write a program for a game – keep monster on screen if its still alive.

```
<h2>JavaScript Loops</h2>
<p>Example of while Loop</p>

<script>
var entity = "Monster";
var life = 10;
while (life != 0) {
    document.write("Keep " + entity + " on screen<br>");
    console.log("Keep " + entity + " on screen ");
    life--;
}
</script>
```

You still see a counter being used. If not, the while loop will run infinitely!

What it looks like when view on the browser:

JavaScript Loops

Example of while Loop

Keep Monster on screen
Keep Monster on screen
Keep Monster on screen
Keep Monster on screen
Keep Monster on screen
Keep Monster on screen
Keep Monster on screen
Keep Monster on screen
Keep Monster on screen
Keep Monster on screen
Keep Monster on screen



How Loops are used

- While **while** loops are not designed to iterate through a range of numbers like **for** loop, it can be used to do the same.

```
<h2>JavaScript Loops</h2>
<p>Example of while Loop</p>

<script>
var i = 0;
while (i <= 10) {
    document.write(i + " ");
    console.log(i + " ");
    i++;
}
</script>
```

What it looks like when view on the browser:

JavaScript Loops

Example of while Loop

0 1 2 3 4 5 6 7 8 9 10



How Loops are used

- Let's look at the basic syntax of **do while** loop

```
//do while
do {
    //code block
} while (condition)
```

- Explain:

- **do while** loop executes the code block at least once and continues to run as long as the condition is *true*
- loop ends if condition evaluates to *false*

- Common use:

- When a task is required to be executed once before the conditions are evaluated.



How Loops are used

- Example of **do while** loop: using the same game program – keep monster on screen if its still alive.

```
<h2>JavaScript Loops</h2>
<p>Example of do while Loop</p>

<script>
//do while
var entity = "Monster";
var life = 10;

do {
    document.write("Keep " + entity + " on screen<br>");
    console.log("Keep " + entity + " on screen ");
    life--;
} while (life != 0);
</script>
```

As in the while loop, you also see a counter being used here. If not, the do while loop will run infinitely!

What it looks like when view on the browser:

JavaScript Loops

Example of do while Loop

Keep Monster on screen
Keep Monster on screen
Keep Monster on screen
Keep Monster on screen
Keep Monster on screen
Keep Monster on screen
Keep Monster on screen
Keep Monster on screen
Keep Monster on screen
Keep Monster on screen
Keep Monster on screen



How Loops are used

- In this **do while loop** example, it shows how the loop will execute the code block at least once even the condition is false at the start of the program.

```
<h2>JavaScript Loops</h2>
<p>Example of do while Loop</p>

<script>
//do while
var i = 11;

do {
    document.write(i + " ");
    console.log(i + " ");
    i++;
} while (i <= 10);
</script>
```

What it looks like when view on the browser:

JavaScript Loops

Example of do while Loop

11

Arrays



How Arrays are used

- The basic syntax of array:

This declare/create a new empty array named arrayName

```
//Array  
var arrayName = [];
```

This declare/create a new array named arrayName that contains a list of data

```
//Array  
var arrayName = [value1, value2, value3, ....];    or  
//Array  
var arrayName = new Array(value1, value2, value3, ....);
```

Examples:

```
//Array  
var fruits = ["apples", "oranges", "pears"];  
//Array  
var scores = [78, 85, 91];
```



How Arrays are used

- Array Methods:

- These are common methods in JavaScript array to manipulate its data:

- `push()` - adds a new item at the end of the array list
 - `pop()` - removes the last item of the array list, returns the item removed
 - `shift()` - same as pop but the first item of the array list
 - `unshift()` - same as push but at the beginning of the array list
 - `splice()` - inserts new item to and also removes item on the array list
(specify position to insert, how many to remove, new items)
 - `slice()` - slice part of array list
(specify index position to start slicing)
 - `sort()` - sorts an array list alphabetically
 - `reverse()` - reverse the order of array list
 - `concat()` - merging arrays



How Arrays are used

- Examples of array methods

- We will use the fruits array:

```
//Array  
var fruits = ["apples", "oranges", "pears"];
```

Code:

- `push()`
`fruits.push("grapes");`
`document.write(fruits);`
- `pop()`
`fruits.pop();`
`document.write(fruits);`
- `shift()`
`fruits.shift();`
`document.write(fruits);`
- `unshift()`
`fruits.unshift("apples");`
`document.write(fruits);`

View on the browser:

apples,oranges,pears,grapes

apples,oranges,pears

oranges,pears

apples,oranges,pears



How Arrays are used

- Examples of array methods (con't)

- We will use the fruits array:

```
//Array  
var fruits = ["apples", "oranges", "pears"];
```

Code:

- splice()

```
fruits.splice(1,0,"grapes","kiwi");  
document.write(fruits);
```
- slice()

```
var favFruits = fruits.slice(2);  
document.write(favFruits);
```
- sort()

```
fruits.sort();  
document.write(fruits);
```
- reverse()

```
fruits.reverse();  
document.write(fruits);
```

View on the browser:

apples,grapes,kiwi,oranges,pears

kiwi,oranges,pears

apples,grapes,kiwi,oranges,pears

pears,oranges,kiwi,grapes,apples



How Arrays are used

- Examples of array methods (con't)

- We will use 2 new arrays:

```
//Arrays  
var beenThereList = ["New York City","London","Rome"];  
var bucketList = ["Shanghai","Santiago"];
```

- concat()

Code:

```
var myList = beenThereList.concat(bucketList);  
document.write(myList);
```

View on the browser: New York City,London,Rome,Shanghai,Santiago

Popup Boxes



JavaScript Popup Boxes

• Popup Types

- JavaScript provides different built-in functions to display popup messages for different purposes, and they are:
 - Alert Box
 - Confirm Box
 - Prompt Box



JavaScript Popup Boxes

- Alert Box

- Use alert() function to display a popup message to the user. This popup will have **OK** button to close the popup.

- Syntax:

```
alert();
```

- Examples:

```
<h1>Demo: alert()</h1>
<script>
    alert("This is alert box!"); // display string message

    alert(100); // display number

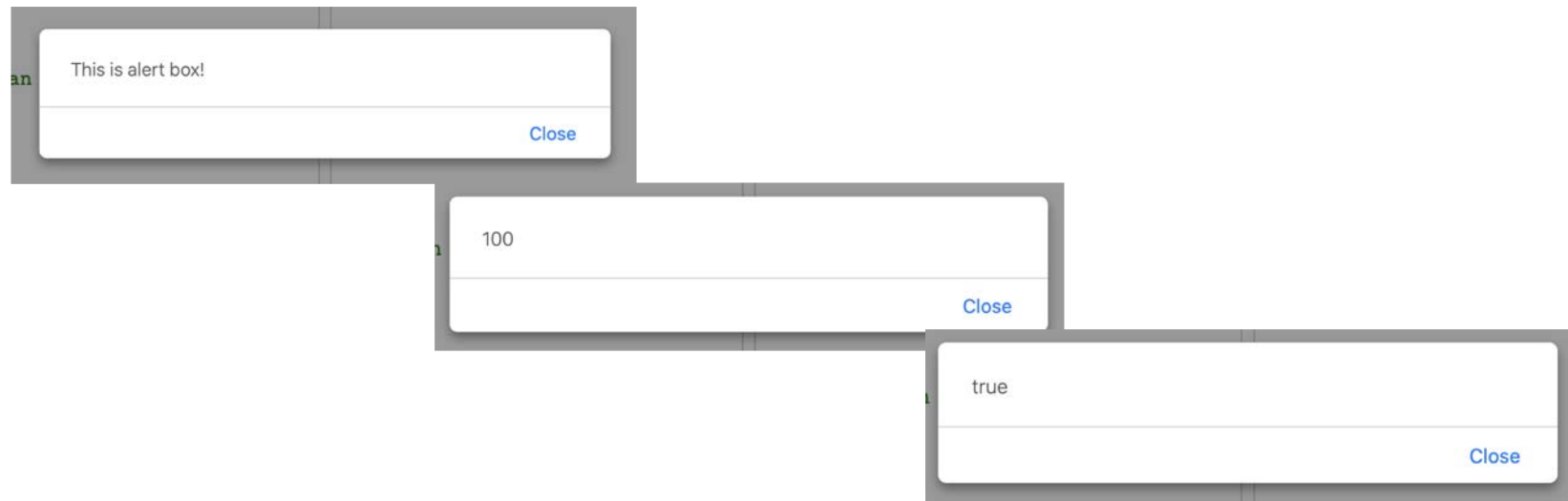
    alert(true); // display boolean
</script>
```



JavaScript Popup Boxes

- Alert Box

- When the page loads on the browser, a alert box will appear each time the button is click to dismiss it before the next alert box appears. The appearance of the alert boxes goes by the order they were coded starting from the top.





JavaScript Popup Boxes

• Confirm Box

- Use `confirm()` function to take the user's confirmation before allowing them to proceed. For example, saving updated data or deleting existing data. This function comes with two buttons, **OK** and **Cancel**. You can check which button the user has clicked and proceed accordingly.

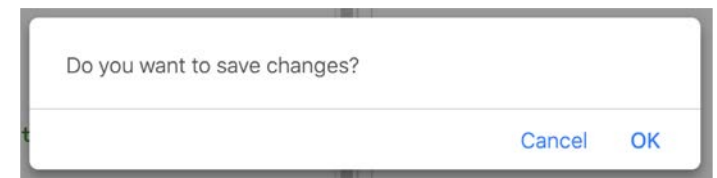
- Syntax:

```
confirm();
```

- Example:

```
<script>  
    confirm("Do you want to save changes?");  
</script>
```

View on Browser:





JavaScript Popup Boxes

- Prompt Box

- Use prompt() function to take the user's input to do further actions on a web page. For example, calculating interest based on users' preferred loan period. Prompt function takes two string parameters. First parameter is the message to be displayed and second parameter is the default value which will be in input text when the message is displayed. The prompt function also comes with 2 buttons, **OK** and **Cancel**.

- Syntax:

```
prompt([string message], [string defaultValue]);
```

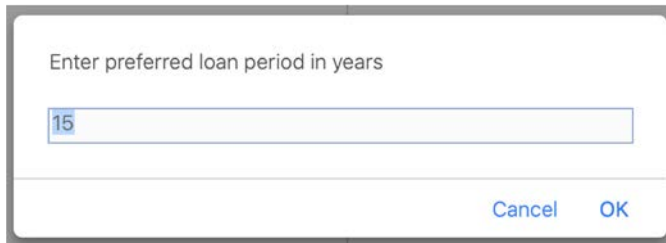


JavaScript Popup Boxes

- Example:

```
<script>  
    prompt("Enter preferred loan period in years", "15");  
</script>
```

- View on Browser:



You can also leave the default message out:

```
<script>  
    prompt("Enter preferred loan period in years");  
</script>
```

Resources

<https://www.javascripttutorial.net/>

<https://www.tutorialrepublic.com/javascript-tutorial/>

<https://developers.google.com/web/tools/chrome-devtools/console/javascript>

Questions?