



Codergirl - JavaScript

Class 2

March 10th, 2020

# Agenda

- Studio review
- Conditionals and Debugging
- Recap of Exercises
- Studio

# boolean

true or false

NOT True or False and

NOT ‘true’ or ‘false’

let ateDinner = true;

let readyToLearn = true;

let bored = false;

# Comparison Operators

- `==`
- `!=`
- `>`
- `<`
- `>=`
- `<=`

## Comparison Operators

Operator	Description	Examples Returning <b>true</b>	Examples Returning <b>false</b>
Equal (==)	Returns <b>true</b> if the two operands are equal, and <b>false</b> otherwise.	<code>7 == 7</code> <code>"dog" == "dog"</code>	<code>7 == 5</code> <code>"dog" == "cat"</code>
Not equal (!=)	Returns <b>true</b> if the two operands are not equal, and <b>false</b> otherwise.	<code>7 != 5</code> <code>"dog" != "cat"</code>	<code>7 != 7</code> <code>"dog" != "dog"</code>
Greater than (>)	Returns <b>true</b> if the left-hand operand is greater than the right-hand operand, and <b>false</b> otherwise.	<code>7 &gt; 5</code> <code>'b' &gt; 'a'</code>	<code>5 &gt; 7</code> <code>'a' &gt; 'b'</code>
Less than (<)	Returns <b>true</b> if the left-hand operand is less than the right-hand operand, and <b>false</b> otherwise.	<code>5 &lt; 7</code> <code>'a' &lt; 'b'</code>	<code>7 &lt; 5</code> <code>'b' &lt; 'a'</code>
Greater than or equal (>=)	Returns <b>true</b> if the left-hand operand is greater than or equal to the right-hand operand, and <b>false</b> otherwise.	<code>7 &gt;= 5</code> <code>7 &gt;= 7</code> <code>'b' &gt;= 'a'</code> <code>'b' &gt;= 'b'</code>	<code>5 &gt;= 7</code> <code>'a' &gt;= 'b'</code>
Less than or equal (<=)	Returns <b>true</b> if the left-hand operand is less than or equal to the right-hand operand, and <b>false</b> otherwise.	<code>5 &lt;= 7</code> <code>5 &lt;= 5</code> <code>'a' &lt;= 'b'</code> <code>'a' &lt;= 'a'</code>	<code>7 &lt;= 5</code> <code>'b' &lt;= 'a'</code>

# Loose/Strict Equality

- **== versus ===**

<code>4 == "4"</code>	true
<code>4 === "4"</code>	false
<code>0 == ""</code>	true
<code>0 === ""</code>	false

Use a === strict equality (or !== strict inequality) comparisons to compare values and avoid type conversion issues:

# Logical Operators

- `&& || !`
- `homeworkCompleted && choresCompleted`
- `ateFruits || ateVeggies`
- `!misbehaving`

# Truth tables for && and ||

For && to be true all conditions should be true

For || to be true any one of the conditions should be true

## Operator Precedence

Precedence	Category	Operators
(highest)	Logical NOT	!
	Exponentiation	**
	Multiplication and division	*, /, %
	Addition and subtraction	+,-
	Comparison	<=, >=, >, <
	Equality	==, !=, ==, !=
	Logical AND	&&
(lowest)	Logical OR	

END TABLE

# if statements

```
if(condition) {
```

```
    do X
```

```
}
```

**if else**

```
if(condition) {  
    do X  
} else {  
    do Y  
}
```

if else if

if(condition) {

    do X

} else if(condition) {

    do Y

}

## nested if

```
if(condition A) {  
    if(condition B) {  
        do X  
    }  
}
```

# Errors

- Syntax - error during parsing.
- Runtime - error during execution.
- Logic – unexpected outcome.

## JavaScript Error Types

Error Type	Description	Example of code triggering the error	Example description
<b>SyntaxError</b>	Occurs when trying to parse syntactically invalid code.	<code>console.log("hello";</code>	The call to <code>console.log</code> does not have a required close parenthesis.
<b>ReferenceError</b>	Occurs when a non-existent variable is used/referenced.	<code>1   let firstName = "Jack"; 2   console.log(firstname);</code>	The variable <code>firstname</code> does not exist; it is a misspelling of <code>firstName</code> .
<b>TypeError</b>	Occurs when trying to use a value in an invalid way.	<code>1();</code>	The numeric value <code>1</code> is not a function, so trying to use it as one results in <b>TypeError: 1 is not a function</b> .
<b>RangeError</b>	Occurs when passing an invalid value to a function.	<code>let nums = Array(-1);</code>	The constructor function <code>Array(n)</code> creates an empty array of length <code>n</code> . It is not possible to create an array with negative length, so the code results in <b>RangeError: Invalid array length</b> .
<b>URIError</b>	Occurs when improperly using a global URI-handling function. ('URI' = Uniform Resource Identifier)	<code>decodeURI('%');</code>	The <code>%</code> character is used to encode characters not otherwise allowed in URLs, such as spaces ( <code>%20</code> ). If an invalid character encoding is given, a <b>URIError</b> results.
<b>Error</b>	The type from which all other errors are built. It can be used to generate programmer-triggered and programmer-defined errors.	<code>throw Error("Something bad happened!");</code>	Manually triggers an error with the given message.

Each time you encounter a new error type, take the time to understand what it is, and what JavaScript is trying to tell you.

Remember, **error messages are your friends!**

# Debugging

- error messages
- console.log
- google

# Best practices

- Tests
- Incremental changes
- Pairing/Mobbing



# Code examples and exercises

# Questions?

# Studio time!