Javascript: Boolean Expressions

Boolean Logic

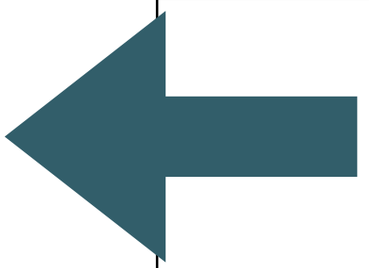
- Write conditional logic using boolean operators
- List all of the falsey values in JavaScript
- Use if/else and switch statements to include conditional logic in your JavaScript code
- Explain the difference between == and === in JavaScript
- Convert between data types explicitly in JavaScript

Conditional Logic

- An essential part of writing programs is being able to execute code that depends on certain conditions. For example:
    - You want the navigation bar on your website to look different based on whether or not someone is logged in
    - If someone enters their password incorrectly, you want to let them know; otherwise, you want to log them in
    - You're building a tic-tac-toe game, and want to know whether it's X's turn or O's turn
    - You're building a social network and want to keep person A from seeing person B's profile unless the two of them are friends

```
var instructor = 'Brenda';

// we begin with an "if" statement
// followed by a condition in ()
// and a block of code inside of {}
if (instructor === 'Brenda') {
  console.log('Yes!');
} else {
  console.log('No');
}
```



Always true

- Notice that we used a === instead of =.
- Anytime that we use more than one equals operator (we can either use == or ===) we are doing a comparison (comparing values).
- When we use a single equals operator =, we are doing an assignment (setting a variable equal to some value).

```
var favoriteFood = prompt('What\'s your favorite food?');

if (favoriteFood === 'pizza') {
  console.log('Woah! My favorite food is pizza too!');
} else {
  console.log('That\'s cool. My favorite food is pizza.');
}
```

- In this version, the boolean expression will be true/false depending on the value entered in ‘prompt’

Difference between “==“ and “===“

- Two different operators for comparison: the double and triple equals.
- Both operators check whether the two things being compared have the same value, but there's one important difference.
  - == allows for ***type coercion*** of the values,
  - === does not.
- To understand the difference between these operators, we first need to understand what is meant by ***type coercion***.

Type Coercion 1

- Add a number and a string.

- In a lot of programming languages, this would throw an error, but JavaScript is more accommodating

```
5 + 'hi'; // '5hi'
```

- It evaluates the expression 5 + "hi" by first coercing 5 into a string, and then interpreting the "+" operator as string concatenation.

- So it combines the string "5" with the string "hi" into the string "5hi"

Type Coercion 2

- JavaScript expects the values inside of parentheses that come after the keyword **if** to be booleans.

- If you pass in a value which is not a boolean, JavaScript will coerce the value to a boolean according to the rules for ***truthy/falsey*** values (more on this later)

```
if ('foo') {
  console.log('this will show up!');
}

if (null) {
  console.log('this won\'t show up!');
}
```

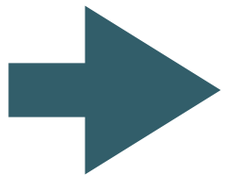
Type Coercion 3

- A very common way to coerce a stringified number back into a number.

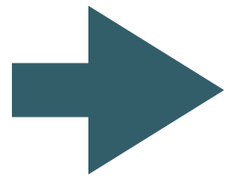
- By prefacing the string with the plus sign, JavaScript will perform a coercion on the value and convert it from a string value to a number value.

```
+'304'; // 304
```

“==“ Vs “===“ again

==  loose

```
5 == '5'; // true
'true' == true; // false
true == 1; // true
undefined == null; // true
```

===  strict

```
5 === '5'; // false
'true' === true; // false
true === 1; // false
undefined === null; // false
```

- == allows for coercion while === doesn't.
- *If you don't want to have to think about coercion in your comparisons, stick to ===.*

```
var x = 4;
if (x <= 5) {
  console.log('x is less than or equal to five!');
} else {
  console.log('x is not less than or equal to five!');
}
```

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
===	is equal to (checks for both value and type)	x=5 y="5" x==y returns true x===y returns false
!=	is not equal	5!=8 returns true
!==	Is not identical	4!==5 (true) 5!==5 (false) 5!=='5' (true)
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Falsey Values

- Some values (aside from false) are actually false as well, when they're used in a context where JavaScript expects a boolean value

- Even if they do not have a "value" of false, these values will be translated (or "coerced") to false when evaluated in a boolean expression.

6 Falsey Values in Javascript

```
0
""
null
undefined
false
NaN // (short for not a number)
```

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

If-Else

- Sometimes you may have more than two conditions to check.
- In this case, you can chain together multiple conditions using else

```
if (number >= 1000) {
  console.log('Woah, thats a big number!');
} else if (number >= 0) {
  console.log('Thats a cool number.');
} else {
  console.log('Negative numbers?! Thats just bananas.');
}
```

Switch

- Another way to write conditional logic is to use a switch statement.
- While these are used less frequently, they can be quite useful when there are multiple conditions that can be met.
- Notice that each case clause needs to end with a break so that we exit the switch statement.

```
switch (feeling) {
  case 'happy':
    console.log("Awesome, Im feeling happy too!);
    break;
  case 'sa':
    console.log('Thats too bad, I hope you feel better soon.');
    break;
  case 'hungry':
    console.log('Me too, lets go eat some pizza!');
    break;
  default:
    console.log('I see. Thanks for sharing!');
}
```

Modulus Operator

```
5 % 3 === 2 // true (the remainder when five is divided by 3 is 2)

var num = prompt('Please enter a whole number');
if ( num % 2 === 0 ) {
  console.log('the num variable is even!')
} else if ( num % 2 === 1) {
  console.log('the num variable is odd!')
} else {
  console.log('Hey! I asked for a whole number!');
}
```