



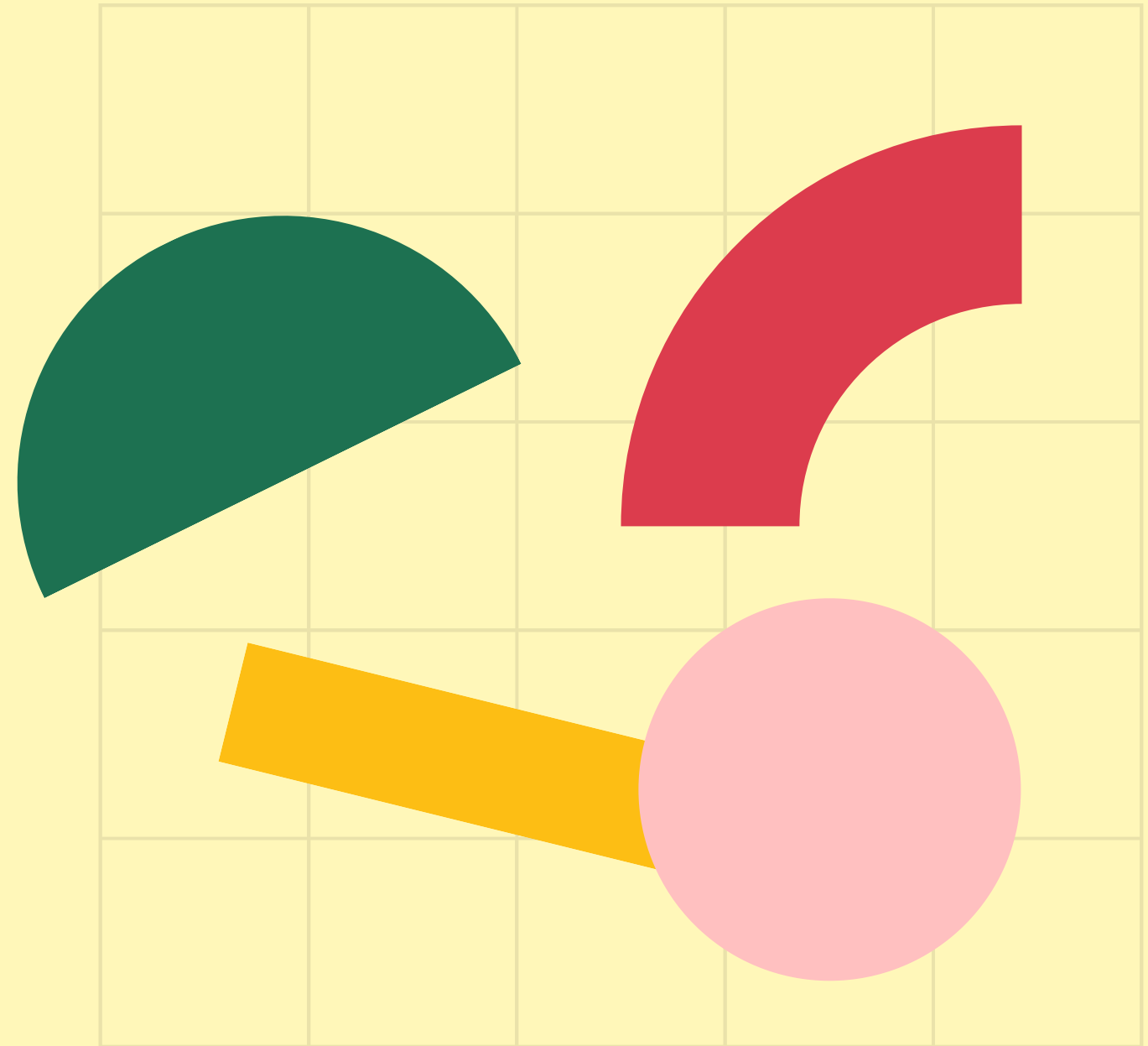
# Booleans

Decision Making with JavaScript

# Unit Goals

what we'll cover

- comparisons
- `console.log()`
- conditionals
- nesting
- logical operators




# COMPARISONS



```
> // greater than  
< // less than  
>= // greater than or equal to  
<= // less than or equal to  
== // equality  
!= // not equal  
=== // strict equality  
!== // strict non-equality
```

# COMPARISONS

## EXAMPLES



```
10 > 1;           //true
0.2 > 0.3;        //false
-10 < 0;          //true
50.5 < 5;         //false
0.5 <= 0.5;       //true
99 >= 4;          //true
99 >= 99;         //true
'a' < 'b';        //true
'A' > 'a';        //false
```

Notice these all return a Boolean!

Though it's uncommon, you can compare strings. Just be careful, things get dicey when dealing with case, special characters, and accents!

==

vs

==

# **== DOUBLE EQUALS**

- Checks for equality of value, but not equality of type.
- It coerces both values to the same type and then compares them.
- This can lead to some unexpected results!

# == EXAMPLES



```
5 == 5;           //true
'b' == 'c';       //false
7 == '7';         //true
0 == '';          //true
true == false;    //false
0 == false;       //true
null == undefined; //true
```

# === TRIPLE EQUALS



```
5 === 5; //true
1 === 2; //false
2 === '2'; //false
false === 0; //false

//Same applies for != and !==
10 != '10'; //false
10 !== '10'; //true
```

CHECK FOR EQUALITY OF VALUE AND TYPE



# console.log()

points arguments to the console  
(needed later when working with files)



# RUNNING CODE FROM A FILE

```
//Put your code in the JS File
alert('Hello from JS!');

//Won't show up!!
"hi".toUpperCase();

//Will show up!
console.log("hi".toUpperCase());
```

write your code in in a  
.js file

```
<!DOCTYPE html>
<html>
<head>
  <title>JS Demo</title>
  <script src="app.js"></script>
</head>
<body>

</body>
</html>
```

include your script in a  
.html file

# CONDITIONALS

## MAKING DECISIONS WITH CODE



# IF STATEMENT

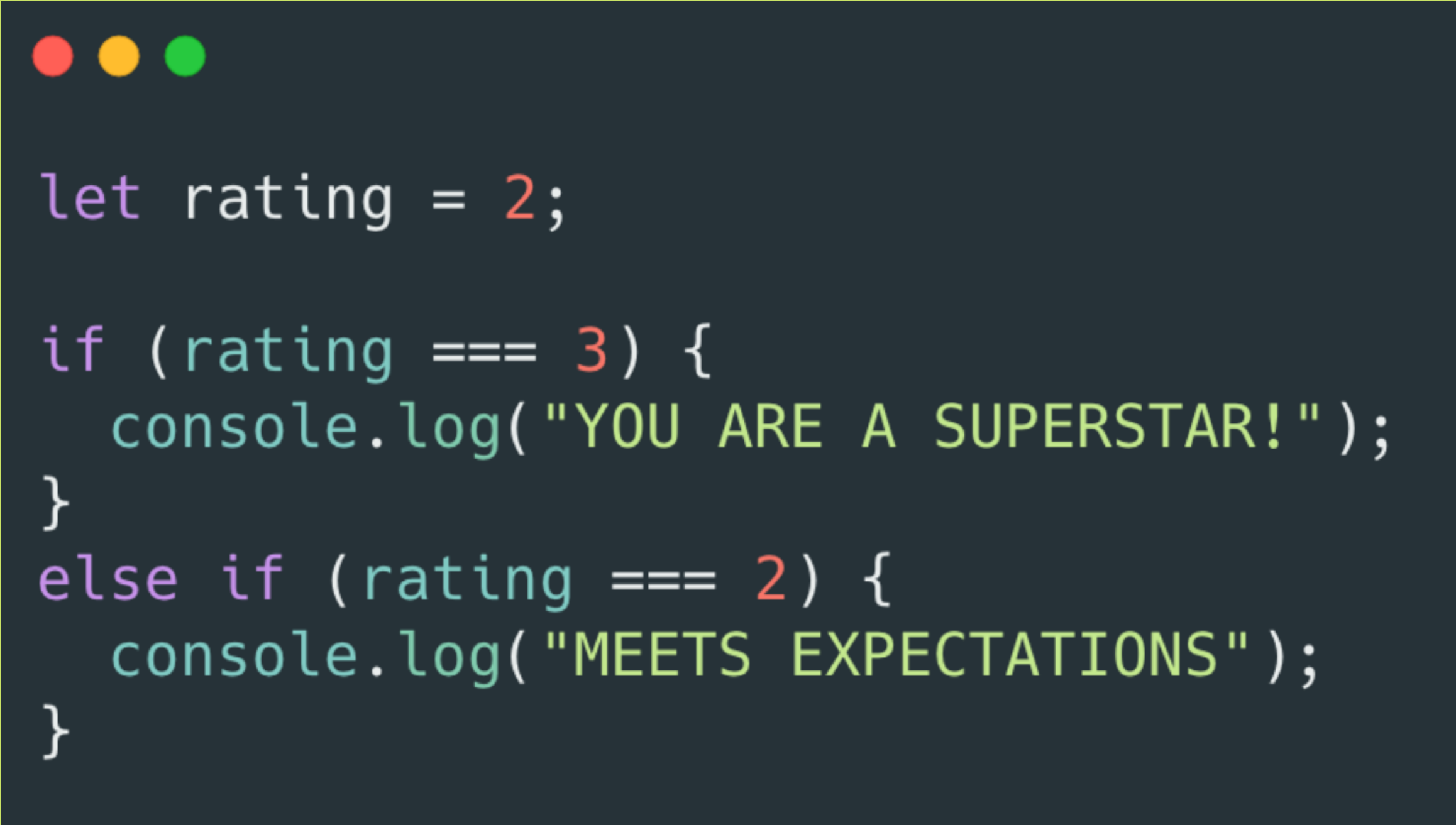
ONLY RUNS CODE IF GIVEN CONDITION IS TRUE



```
let rating = 3;  
  
if (rating === 3) {  
  console.log("YOU ARE A SUPERSTAR!");  
}
```

# ELSE IF

IF NOT THE FIRST THING MAYBE THIS ONE?

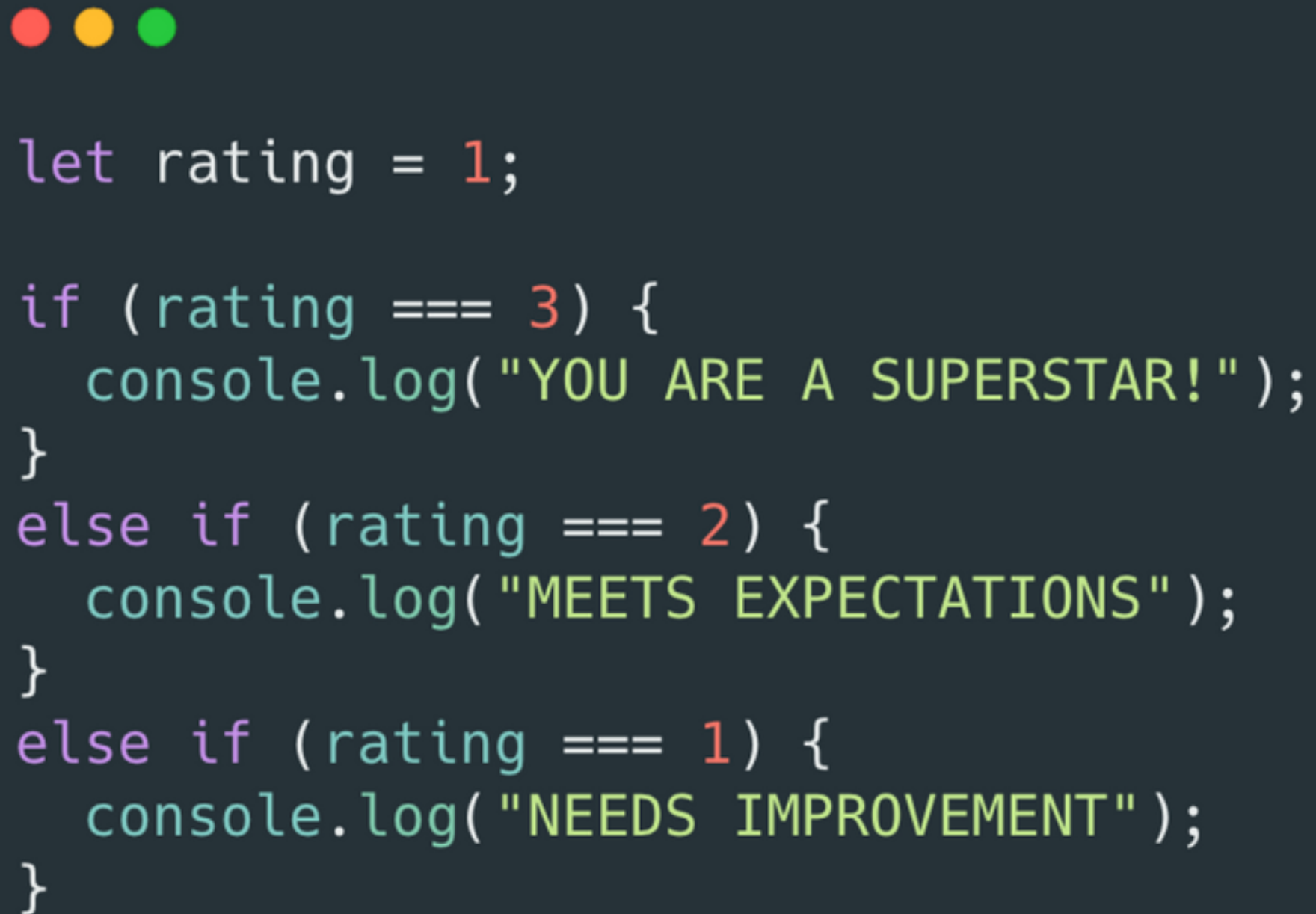


```
let rating = 2;

if (rating === 3) {
  console.log("YOU ARE A SUPERSTAR!");
}
else if (rating === 2) {
  console.log("MEETS EXPECTATIONS");
}
```

# ELSE IF

MULTIPLE ELSE IF'S CAN BE ADDED



```
let rating = 1;

if (rating === 3) {
  console.log("YOU ARE A SUPERSTAR!");
}
else if (rating === 2) {
  console.log("MEETS EXPECTATIONS");
}
else if (rating === 1) {
  console.log("NEEDS IMPROVEMENT");
}
```

# ELSE


IF NOTHING ELSE WAS TRUE, DO THIS

```
let rating = -99;

if (rating === 3) {
  console.log("YOU ARE A SUPERSTAR!");
}
else if (rating === 2) {
  console.log("MEETS EXPECTATIONS");
}
else if (rating === 1) {
  console.log("NEEDS IMPROVEMENT");
}
else {
  console.log("INVALID RATING!");
}
```

# NESTING

WE CAN NEST CONDITIONALS  
INSIDE CONDITIONALS



```
let password = "cat dog";  
if (password.length >= 6) {  
  if (password.indexOf(' ') !== -1) {  
    console.log("Password cannot include spaces");  
  }  
  else {  
    console.log("Valid password!!")  
  }  
}  
else {  
  console.log("Password too short!");  
}
```



# TRUTHY & FALSY VALUES

- All JS values have an inherent truthyness or falsyness about them.
- Falsy values:
  - false
  - 0"" (empty string) null
  - undefined
  - NaN
- Everything else is truthy!



# LOGICAL OPERATORS

COMBINING EXPRESSIONS



**&&**

A diagram of a logical operator box. It consists of a thick black rectangular border. Inside the box, the text "&&" is centered in a bold, black, sans-serif font.



**||**

A diagram of a logical operator box. It consists of a thick black rectangular border. Inside the box, the text "||" is centered in a bold, black, sans-serif font.



**!**

A diagram of a logical operator box. It consists of a thick black rectangular border. Inside the box, the text "!" is centered in a bold, black, sans-serif font.

# AND

BOTH SIDES MUST BE TRUE, FOR THE  
ENTIRE THING TO BE TRUE




```
1 <= 4 && 'a' === 'a'; //true
```

```
9 > 10 && 9 >= 9; //false
```

```
'abc'.length === 3 && 1+1 === 4; //false
```

# AND

BOTH SIDES MUST BE TRUE, FOR THE  
ENTIRE THING TO BE TRUE



```
let password = 'taco tuesday';

if(password.length >= 6 && password.indexOf(' ') === -1){
  console.log("Valid Password!");
}
else {
  console.log("INVALID PASSWORD!");
}
```

# OR

IF ONE SIDE IS TRUE, THE ENTIRE THING IS  
TRUE



```
//only one side needs to be true!
```

```
1 !== 1 || 10 === 10 //true
```

```
10/2 === 5 || null //true
```

```
0 || undefined //false
```

# OR

IF ONE SIDE IS TRUE, THE ENTIRE THING IS  
TRUE



```
let age = 76;
```

```
if(age < 6 || age >= 65){  
  console.log('You get in for free!');  
}  
else {  
  console.log('That will be $10 please');  
}
```

# NOT

!EXPRESSION RETURNS TRUE IF  
EXPRESSION IS FALSE



```
!null //true
```

```
! (0 === 0) //false
```

```
!(3 <= 4) //false
```

# SWITCH

THE SWITCH STATEMENT IS ANOTHER CONTROL-FLOW STATEMENT THAT CAN REPLACE MULTIPLE IF STATEMENTS.

THIS SYNTAX IS KIND OF UNWIELDY AND HARD TO REMEMBER, BUT IT'S GOOD TO KNOW ABOUT!



```
const day = 2;
switch (day) {
  case 1:
    console.log("MONDAY!");
    break;
  case 2:
    console.log("TUESDAY!");
    break;
  case 3:
    console.log("WEDNESDAY");
    break;
  case 4:
    console.log("THURSDAY");
    break;
  case 5:
    console.log("FRIDAY");
    break;
  default:
    console.log("INVALID NUMBER!")
}
```