CONDITIONAL EXECUTION

MAKING DECISIONS

- We can use JavaScript to make decisions
  - Uses Boolean Expressions (an expression that is true or false)
  - If you don’t have a Boolean expression, you can use comparisons to create one
- All data that we use for making decisions must be in a comparable format
  - Strings
  - Booleans
  - Numbers
- Because these are the only things JavaScript knows how to compare

DECISIONS COMPUTERS CAN'T MAKE

- If we can't quantify, "stringify", or "booleanify" our inputs/data, then a computer can't base a decision on that data
  - For Example: Should I eat Cake or Ice Cream?
- The above is a true or false question, but how can a computer make a meaningful decision if the data does not contain enough information to answer the question!
- The computer can't quantify all of the factors that will affect my decision
  - Google: Cannot tell the age of a person from a picture
  - We cannot predict earthquakes far in advance, because we haven't found meaningful data(statistically significant) that can predict it

BOOLEAN EXPRESSIONS

- Comparison operators ===, >=, <= will result in a boolean expression
  - Ex. **`num >= 5;`**
- Logical operators, && and || are how we combine Boolean expressions.
  - Ex. **`num >= 5 && num <= 9;`**

```
let num = 10;
let result = num >= 5 && num <= 9;
```

- So what is stored in result after these two statements execute?

IF STATEMENTS

- If statements are used for conditional execution
- Up till now, we have run our JavaScript top to bottom without skipping a line
- If statements allow us to skip lines of code
- Or to only execute certain lines of code IF some condition has been met

```
if(boolean condition) {
     //Only executed when true
}
```

IF ELSE STATEMENTS

- We can add an else statement
- The else code block will only execute when the if condition evaluates to FALSE

```
if(<boolean condition>) {
   //Only executed when condition is true
} else {
   //only executed when condition is false
}
```

EXAMPLE USING IF ELSE

```
function greeting(hour){
  let greeting = “”;
  if (hour < 18) {
      greeting = "Good day";
  } else {
      greeting = "Good evening";
  }
  return greeting;
}
```

NOTE

- Do not forget that *MOST* lines of JavaScript end in a semi colon

```
    console.log(“Hello World!”);
```

- One of the exceptions to this rule is when we are writing if statements

```
if (hour < 18) {
    greeting = "Good day";
} else {
    greeting = "Good evening";
}
```

CODE BLOCKS

- If statements naturally **only execute the FIRST line** following the if statement

```
if(height > 180)
  alert(“you are above average height”);
```

- So what do we do if we want to execute multiple statements?

CODE BLOCKS

- This is where code blocks come in handy
- A code block is a group of JavaScript statements surrounded by curly braces { }
- If you follow an if statement with a code block, then all statements in the { } will execute

```
if(height > 180){
    alert(“you are tall.”);
    alert(“your height in inches is 70.86“ );
}
```

EXERCISE

- What’s wrong with the following code fragment? Rewrite it so that it produces the correct output. Assume total, max, and sum have already been defined.

```
if(total === max)
      if(total < sum)
            console.log(“The total equals max and is less than sum”);
else console.log(“The total is not equal to max”);
```

QUESTION

- What output is produced by the following code?

```
let num = 87, max = 25;
if (num >= max*2)
    console.log(“apple”);
    console.log(“orange”);
console.log(“banana”);
```

IF, ELSE IF, ELSE

- You can add an else if statement to the traditional if else structure
- **This is used when you have more than two possible cases to execute**

```
if(<condition 1>) {
   //executed if condition 1 is true
} else if(<condition 2>) {
  // executed if condition 1 is false, and condition 2
is true
} else {
  //executed when all conditions are false.
}
```

EXAMPLE USING ELSE IF

```
function greeting(hour){
  let greeting = “”;
  if (time < 10) {
      greeting = "Good morning";
  } else if (time < 20) {
      greeting = "Good day";
  } else {
      greeting = "Good evening";
  }
  return greeting;
}
```

QUESTION

- What’s wrong with the following code?

```
if(length = 100)
    console.log(“The length is 1 meter”);
```

PRACTICE WRITING IT

- At the end of a game, a player’s score determines what message is printed to the screen
  - If the user’s score is 100, the message “Great Success!” is displayed
  - If the user’s score is under 100 but above 50, the message “Good Work!” is displayed
  - If the user’s score is equal to or under 50, the message “Better luck next time!” is displayed
- How do we write this function using if statements?

SOLUTION

```
function checkScore(score){
  if(score === 100)
    return “Great Success”;
  else if (score > 50)
    return “Good Work!”;
  else return “Better luck next time!”;
}
```

BETTER SOLUTION

```
function checkScore(score){
  if(score >= 100)
    return “Great Success”;
  else if (score > 50)
    return “Good Work!”;
  else return “Better luck next time!”;
}
```

NESTING IF STATEMENTS

- For more complex conditions you can nest an if statement inside of an if statement
- So for example, if you wanted to see if someone was wearing socks or not and sandals or runners you could write

NESTING IF STATEMENTS

```
if(socks === true){
     if(shoes == “sandals”)
          alert(“you are a fashion disaster”);
     else alert(“good job”);
}else{
     if(shoes == “runners”)
          alert(“who wears runners with no socks?!”);
     else alert(“good job”);
}
```

ALTERNATIVE TO NESTING IF STATEMENTS

```
if(socks === true && shoes === “sandals”)
        alert(“you are a fashion disaster”);
else if(socks === true && shoes === “runners”)
        alert(“good job”);
else if(socks === false && shoes === “sandals”)
    alert(“good job”);
else alert(“who wears runners with no socks?!”);
```

LOGICAL OR

- Examples using the logical or

```
function isLuckyNumber(input){
   if (input == 5 || input == 25)
      return true;
   else return false;
}
```

a	b	Result of a || b
T	T	T
T	F	T
F	T	T
F	F	F

QUESTION

- What output is produced by the following code?

```
let limit = 100, num1 = 15, num2 = 40;
if (limit <= limit){
    if (num1 === num2)
        console.log("lemon");
    console.log("lime");
}
console.log("grape");
```

VERIFYING INPUT

- When we obtain input form the user we have to make sure it is what we expect
- We can use if statements to verify that the user has entered valid input
- We can check if the something is a number using the isNaN() function

```
isNaN(100) returns false
isNaN(“100”) returns false
isNaN(“MOM”) returns true
isNaN(NaN) returns true
```

EXAMPLE VERIFYING USER INPUT

```
function isValidNumber(num){
   if(isNaN(num))
      return false;
   else return true;
}
```

SWITCH STATEMENTS

- Are like a condensed version of several if, else if, else statements

```
switch(<expression>)
{
     case <expression1>:
         //code block 1
         break;
     case <expression 2>:
        //code block 2
         break;
      default:
        //default code block
         break;
}
```

HOW SWITCH WORKS

- Switch compares the expression in the switch statement, with each of the case expressions using the === operator
- The comparison starts from the top case and continues until it finds one that is true
  - If a true case is not found then the default case will be executed
- However, if no default is specified, and no true cases are found, then no code block is executed.
- The break statements are **necessary**, because switch statements have fall through behavior (i.e., without the break; one case after another would execute)

EXAMPLE

```
let team = prompt(“Enter your favorite avenger!”);
switch(team){
      case “Captain America”:
            alert(“Avengers Assemble!”);
            break;
      case “Iron Man”:
            alert(“Genius. Billionaire. Playboy. Philanthropist.”);
            break;
      case “Hulk”:
            alert(“HULK SMASH!”);
            break;
      case “Thor”:
            alert(“You are not worthy.”);
            break;
      default:
            alert(“You picked Black Widow/Hawkeye/Vision/ScarletWitch”);
}
```

Rewrite the following switch statement using if-else statements ONLY

```
let grade = prompt ("Enter a numeric grade (0 to 100): ");
let category = Math.round(grade / 10);
console.log("That grade is ");
switch (category){
      case 10:
            console.log ("a perfect score. Well done.");
            break;
      case 9:
            console.log ("well above average. Excellent.");
            break;
      case 8:
            console.log ("above average. Nice job.");
            break;
      case 7:
            console.log ("average.");
            break;
      case 6:
            console.log ("below average. You should see the
instructor.");
            break;
      default:
            console.log ("not passing.");

}
```

SOLUTION

```
if(category === 10)
     console.log("a perfect score. Well Done.");
else if (category === 9)
     console.log("well above average. Excellent.");
else if (category === 8)
     console.log("above average. Nice job.");
else if (category === 7)
     console.log("average.");
else if (category === 6)
     console.log("below average. You should see the instructor.");
else console.log("not passing.");
```

NOW YOU TRY IT

- An insurance company has different insurance base rates for different types of cars. The company rate chart is below, implement the switch statement that prints the correct rate based on the vehicle type.

Car Type	Rate
Car	$800
SUV	$1100
Van	$1000
Other	$1200

```
let carType = prompt("Enter car: ");
let rate = 0;
switch(carType)
{
     case "Compact":
         rate = 800;
         break;
     case "Luxury":
        rate = 1200;
         break;
     case "SUV":
        rate = 1100;
         break;
     case "Van":
        rate = 1000;
         break;
     case "Sedan":
        rate = 1150;
         break;
     default:
        rate = 1200;
}
alert("Your insurance rate is: " + rate);
```

OPERATORS REVISITED

- The Unary Operators
  - Effect ONE expression, not two like binary operators do
- !
  - negation
- + and –
  - Positive (or numerical type conversion) and negative
- ++ and --
  - Increment and decrement
  - x++ means x = x + 1;

```
let bool = 15 + 2 > 20;
```

```
let user = +prompt("enter a number.");
user = -user;
```

```
user++;
user--;
```

OPERATORS REVISITED

- ? :
  - The ternary operator
  - Effects THREE expressions
  - Basically a short handed if else statement

```
(<CONDITION>)? //DO IF TRUE : //DO IF FALSE;

let user = prompt(“Enter your gender: ”);
(user === “male”)? alert(“you are male”) : alert(“you are female”);
OR
alert((user === “male”)? “you are male” : “you are female”);
```

EXERCISE

- Write a statement using the ternary operator that checks if a variable called age is less than 50.
- If the age is less than 50 print the statement “You are young!”, otherwise print the statement “You are old!”.

THE REMAINDER OPERATOR

- X % Y will return the remainder after X has been divided by Y
- We can use remainder to check if a number is even or odd

```
function isEven(num){
    if(num % 2 == 0)
        return true;
    else return false;
}
```

EXERCISE

- Write some JavaScript that creates and prints a random phone number of the form XXX–XXX–XXXX
- Include the dashes in the output
- Do not let the first three digits contain an 8 or 9 (but don’t be more restrictive than that)
- Make sure that the second set of three digits is not greater than 742
- Hint: Think through the easiest way to construct the phone number. Each digit does not have to be determined separately.

EXERCISE

- Modify our account program so that the withdrawal and deposit methods check for valid amounts

```
function withdrawal(amount){
    balance = balance - amount;
}

function deposit(amount){
    balance = balance + amount;
}
```