5 Primitive Data Types

```
//Numbers
4
9.3
-10

//String
"Hello World"
"43"

//Booleans
true
false

//Null and undefined
null
undefined
```

```
//Numbers
4
9.3
-10

//We can do some math
4 + 10 //14
1 / 5 //0.2

//Modulo - remainder operator

10 % 3 //1
24 % 2 //0
15%11 //4
```

```
//We can do some math
4 + 10 //14
1 / 5 //0.2

//Modulo - remainder operator

10 % 3 //1
24 % 2 //0
15%11 //4
```

```
//Strings | Single or Double quotes are OK
"Hello, World"
```

```
'Hello, World'

//Concatenation
"charlie" + "brown" //charliebrown

//Escape Characters starts with "\"
"Singin \"Do wah diddy, diddy dum diddy do\""
"This is a backslash: \\"

//Strings have a length property
"hello world".length //11

//Access individual characters using [] and an index
"hello"[0] //"h"
"hello"[4] //"o"
```

Variables

Naming convention: camel case

```
//Variables are simply containers that store values
//They follow this pattern

var yourVariableName = yourValue;

//They can tore all of the values we've seen
var name = "Aldrin";
var secretNumber = 73;
var isAdorable = true;
```

```
//We can also update existing variables
var name = "AJ";
name = "Aldrin";
```

Null and Undefined

```
//The two other primitives are null and undefined

//Variables that are declared, but not
//Initialzed are undefined
//The following variables are undefined:
var name;
var age;

//null is "explicitly nothing"
var currentPlayer = "charlie";
currentPlayer = null; //game over
```

Useful Built-In Methods

```
//alert
//pop-up message
alert("Hello there!!!")

//console.log
//print something to JS console
console.log("Hello from the console!")

//prompt
//to get input from the user
prompt("What is your name?")
var userName = prompt("Please enter your username");
```

3 JavaScript conditional keywords

If
Else If
Else

```
var age = 27;

if (age < 18){
  console.log("Sorry, you are not old enough to enter the venue");
}

else if (age < 21){
  console.log("You can enter, but cannot drink");
}

else {
  console.log("Come on in. You can drink.");
}
```

Simple Guessing Game

```
//Create secretNumber
var secretNumber = 4;

//Ask user for a guess
var guess = prompt("Guess a number");

//check if guess is right
if (Number(guess) === secretNumber) {
  alert("You guessed the right number");
}

else if (guess > secretNumber) {
```

```
  alert("Your guess is too high.");
}

else {
  alert("Your guess is too low.");
}
```

Loops
Repeating Things

DRY – Don't Repeat Yourself

While Loops
Repeat code WHILE a condition is TRUE
Printing number from 1-5

```
var count = 1;

while (count < 6) {
  console.log("count is: " + count);
  count++;
}
```

Printing each character in a string

```
//String we're looping over:
var str = "hello";

//First character is at index 0
var count = 0;

while (count < str.length) {
  console.log(str[count]);
  count++;
}
```

Infinite Loops occur when the terminating condition in a loop is never true

```
var count = 0;

while (count < 10) {
  console.log(count);
}
```

For Loops
Another type of loop

```
//Printing numbers from 1-5 with a for loop
```

```
for (var count = 1; count < 6; count++) {
  console.log(count);
}
```

```
//Printing each character in a string with a for loop
var str = "Hello";

for (var i = 0; i < str.length; i++) {
  console.log(str[i]);
}
```

JS Functions
A fundamental Building Block
Naming convention: camelCase

6 most common naming conventions
-camelCase
-PascalCase
-SCREAMINGCASE
-lazycase
-kebab-case
-snake_case

Functions let us wrap bits of code up to REUSABLE packages. They are one of the building blocks of JS.

```
//declare a function first:
function doSomething() {
  console.log("Hello, World!");
}
```

```
//Suppose I want to write code to sing "Twinkle Twinkle Little Star"
console.log("Twinkle, twinkle, little star,");
console.log("How I wonder what you are!");
console.log("Up above the world so high,");
console.log("Like a diamond in the sky.");

//To sing in again, I have to rewrite all the code. This is not DRY!
```

```
//We can write a function to help us out
function singSong() {
  console.log("Twinkle, twinkle, little star,");
  console.log("How I wonder what you are!");
  console.log("Up above the world so high,");
  console.log("Like a diamond in the sky.");
}

//To sing the sing, we just need to call singSong();
```

```
singSong();
singSong();
singSong();
singSong();
```

Arguments

```
//often we want to write functions that take inputs
function square(num) {
  console.log(num * num);
}
```

```
function sayHello(name) {
  console.log("Hello there, " + name + "!");
}

sayHello("AJ");
```

```
//functiuons can have as many arguments as needed

function area(length, width) {
  console.log(length * width);
}

area(9, 2);

function greet(person1, person2, person3) {
  console.log("hi, " + person1);
  console.log("hi, " + person2);
  console.log("hi, " + person3);
}

greet("AJ", "Aldrin", "Travis");
```

The Return Keyword

Often we want a function to send back an output value

```
function square(x) {
  console.log(x*x);
}

square(4);

"4 squared is: " + square(4);
```

```
function square(x) {
  return x * x;
```

```
}

square(4);
"4 squared is: " + square(4);
```

```
//we use the return keyword to output a value from a function
//this function capitalizes the first char in a string:
function capitalize(str) {
  return str.charAt(0).toUpperCase() + str.slice(1);
}

var city = "paris";
var capital = capitalize(city);
```

```
//the return keyword stops execution of a function
function capitalize(str) {
  if (typeof str === "number") {
    return "that's not a string!"
  }
  return str.charAt(0).toUpperCase() + str.slice(1);
}

var city = "paris";
var capital = capitalize(city);

var num = 37;
var capital = capitalize(num);
```

```
//Another Syntax
//Function Declaration vs. Function Expression

//function declaration
function capitalize(str) {
  return str.charAt(0).toUpperCase() + str.slice(1);
}

//function expression
var capitalize = function(str) {
  return str.charAt(0).toUpperCase() + str.slice(1);
}
```