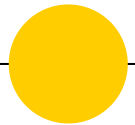
# CS4220 Node.js & Vue.js

Cydney Auman
Albert Cervantes
CSULA



## Intro to Javascript

# Javascript

JavaScript is a cross-platform, lightweight, interpreted, prototype-based object-oriented language with first-class functions. It is a multi-paradigm programming language.

In true object-oriented programming - first a Class is created to serve as a “blueprint” and then objects are created based on this blueprint. Because Javascript is prototype-based object-oriented - a blueprint is never really created.

Javascript shares concepts from functional programming in that it supports first class functions. However, a key component to functional programming is data immutability. In JavaScript, numbers, strings and booleans are immutable. However, objects and arrays are mutable.

# Arrays

JavaScript provides a data type specifically for storing sequences of values.  An array is written as a list of values between square brackets, separated by commas.

```
const alpha = ['a', 'b', 'c', 'd']
```

Elements in the array can be accessed by index. The first index of an array is zero.  So the first element is retrieved with `alpha[0]` which return the value `'a'`.

Javascript arrays also have a length property.  This tells us how many elements it has inside the array. `alpha.length` will return 4.

# Array Methods

**push(***value***)** method adds one or more elements to the **end** of an array and returns the new length of the array.

**pop()** method removes the **last** element from an array and returns that element. This method changes the length of the array.

**unshift(***value***)** method adds one or more elements to the **beginning** of an array and returns the new length of the array.

**shift()** method removes the **first** element from an array and returns that element. This method changes the length of the array.

# Iterating Over Arrays

Loops offer a quick and simple way to perform some action or actions repeatedly. Arrays can be looped over using a standard `for loop`. Or using the `forEach` method.

```
const arr = ['hello', 'world', '!']
for (let i = 0; i < arr.length; i++) {
     console.log(arr[i])
}

arr.forEach(word => {
    console.log(word)
}
```

# Objects

Values of the type object are arbitrary collections of properties, and we can add or remove these properties as we please. One way to create an object is by using a curly brace notation.

```
const transformer = {
    name: 'Optimus Prime',
    team: 'Autobots',
    colors: ['red', 'blue', 'silver']
}
```

# Objects

## Adding Properties

```
transformer.homeWorld = 'Cybertron'
transformer['vehicle'] = 'truck'
```

## Accessing Properties

```
console.log(transformer.homeWorld)  // 'Cybertron'
console.log(transformer['vehicle']) // 'truck'
```

# Object Methods

**Object.keys(***obj***)** returns an array of a given object's properties.

**Object.values(***obj***)** returns an array of a given object's property values.

**Object.assign(***obj, obj, ....***)** method is used to copy the values of all properties from one or more source objects to a target object. It will return the target object.

# Object Looping

A `for...in` loop only iterates over properties in the Object.

```
for (const prop in transformer) {
   console.log(prop)                // property name
   console.log(transformer[prop]) // value
}
```

# Functions

**JavaScript Functions are First-Class Objects**

- They can be assigned to variables, array entries, and properties of other objects.
- They can be passed as arguments to functions.
- They can be returned as values from functions.

# Functions

**JavaScript Functions are composed of four parts:**

- The function keyword.
- An optional name that, if specified, must be a valid JavaScript identifier.
- An optional comma-separated list of parameter names enclosed in parentheses.
- The body of the function, as a series of JavaScript statements enclosed in braces.

```
function addTwo(n,m) {
  return n + m
}
```

# Anonymous Functions

The function below is an **anonymous function** (a function without a name).
Functions stored in variables, do not need names. They are always invoked/called using the variable name.

```
const addThree = function(n){
  return n + 3
}
```

# ES6 Arrow Functions

Arrow functions are functions defined with a new ES6 syntax that uses an “arrow” (=>).

An arrow function expression has a shorter syntax than a function expression and does not bind its own *this*, *arguments*, *super*, or *new.target*. Arrow functions are always anonymous.

```
const addThree = function(n) {
    return n + 3
}

// equivalent to:
const addThree = (n) => {
    return n + 3
}


// equivalent to:
const addThree = n => n + 3
```

# References and Reading

Mozilla Developer Network (Methods and Properties on Array and Objects)

-- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects

Eloquent Javascript

-- http://eloquentjavascript.net/

-- Chapters 4, 5, 6