JavaScript Objects in Detail

* Javascript have one complex data type and five simple data type (primitive), which data are mutable?

What is an Object?

* What’s an object?
* Property of object can be a number? How to call it?

Reference Data Type and Primitive Data Types

Object Data Properties Have Attributes

* 3 attributes all object always have?

Creating Objects

* Object Literals (={ })
* Object Constructor (like function)

Practical Patterns for Creating Objects

* What is the pattern?
* Constructor Pattern for Creating Objects (like function, can input element)
* Prototype Pattern for Creating Objects (like prototype, defined and call)

Further Reading

* How to Access Properties on an Object
  + Dot Notation
  + Bracket Notation (must using if property is number)
* Own and Inherited Properties
  + Using ‘in’ operator to check property exists on an object
* asOwnProperty
* Accessing and Enumerating Properties on Objects
* Accessing Inherited Properties
* Object’s Prototype Attribute and Prototype Property
* Deleting Properties of an Object
* Serialize and Deserialize Objects

JavaScript Prototype in Plain Language

**two interrelated concepts with prototype in JavaScript**

* First, every JavaScript function has a prototype property
* The second concept with prototype in JavaScript is the prototype attribute

What’s the Constructor?

Prototype Attribute of Objects Created with new Object () or Object Literal

**Prototype Attribute of Objects Created With a Constructor Function**

* two general ways an object’s prototype attribute

**Why is Prototype Important and when is it Used?**

* **Prototype Property: Prototype-based Inheritance**
* **Prototype Attribute: Accessing Properties on Objects**

**Object.prototype Properties Inherited by all Objects**

**Additional Information**

**JavaScript Variable Scope and Hoisting Explained**

**Variable Scope**

**Local Variables (Function-level scope)**

* **If You Don’t Declare Your Local Variables, Trouble is Nigh**
* **Local Variables Have Priority Over Global Variables in Functions**

**Global Variables**

* outside a function
* not using “var” keyword
* **setTimeout Variables are Executed in the Global Scope**
* **Do not Pollute the Global Scope**

**Variable Hoisting**

**Function Declaration Overrides Variable Declaration When Hoisted**

**Understand JavaScript Closures With Ease**

**What is a closure?**

**A Basic Example of Closures in JavaScript:**

**A Classic jQuery Example of Closures:**

**Closures’ Rules and Side Effects**

1. **Closures have access to the outer function’s variable even after the outer function returns:**
2. **Closures store references to the outer function’s variables**
3. **Closures Gone Awry**

To fix this side effect (bug) in closures, you can use an **Immediately Invoked Function Expression** (IIFE)

Understand JavaScript Callback Functions and Use Them

**What is a Callback or Higher-order Function?**

**How Callback Functions Work?**

**Callback Functions Are Closures**

**Basic Principles when Implementing Callback Functions**

**Use Named OR Anonymous Functions as Callbacks**

**Pass Parameters to Callback Functions**

**Make Sure Callback is a Function Before Executing It**

**Problem When Using Methods With The**this**Object as Callbacks**

**Use the Call or Apply Function To Preserve**this

**Multiple Callback Functions Allowed**

**“Callback Hell” Problem And Solution**

**Make Your Own Callback Functions**

# Understand JavaScript’s “this” With Clarity, and Master It

**JavaScript’s this Keyword Basics**

**The Biggest Gotcha with JavaScript “this” keyword**

**The use of this in the global scope**

**When *this* is most misunderstood and becomes tricky**

**A bit about “Context” before we continue**

1. **Fix this when used in a method passed as a callback**
2. **Fix this inside closure**
3. **Fix this when method is assigned to a variable**
4. **Fix this when borrowing methods**

# JavaScript’s Apply, Call, and Bind Methods are Essential for JavaScript Professionals

**JavaScript’s Bind Method**

**JavaScript’s Bind Allows Us to Set the this Value on Methods**

**Bind () Allows us to Borrow Methods**

**JavaScript’s Bind Allows Us to Curry a Function**

**JavaScript’s Apply and Call Methods**

**Set the this value with Apply or Call**

**Borrowing Functions with Apply and Call (A Must Know)**

* **Borrowing Array Methods**
* **Borrowing String Methods with Apply and Call**
* **Borrow Other Methods and Functions**

**Use Apply () to Execute Variable-Arity Functions**

# Learn HTML5, CSS3, and Responsive WebSite Design in One Go

**Introduction and Overview**

**Design Strategies and Tools for Responsive UI Design and Layout**

**Developing Responsive, Fluid Layout with Percentages and CSS3 Media Queries**

**Core HTML5 (Semantic Elements, Audio, and Video)**

**All the CSS3 You Need to Know**

**Modern Forms with HTML5 and CSS3**

**Cross-browser HTML5 with HTML5 Boilerplate and Modernizr**

**Twitter Bootstrap 3.0 Responsive Website Layout Tutorial**

# OOP In JavaScript: What You NEED to Know

**Encapsulation and Inheritance Overview**

**Encapsulation in JavaScript**

**Implementation of Combination Constructor/Prototype Pattern**

**Make Instances of the User function**

**Explanation of Combination Constructor/Prototype Pattern**

* **JavaScript Prototype**
* **Constructor Property**
* **Prototype Methods**

**Inheritance in JavaScript**

**Why Inheritance?**

**Implementing the Parasitic Combination Inheritance Pattern**

**Prototypal Inheritance by Douglas Crockford**

* **Object.create method**
* **Why did we manually set the copyOfParent.constructor?**
* **Back to the fun stuff: Creating our quiz OOP style.**
* **The Question Constructor (Parent of all Question Objects):**
* **Add Prototype Methods to The Question Object**
* **Child Questions (Sub Classes of the Question object)**
* **A Drag and Drop Question**

### Further Reading