**UI Areas**

         Concept of Namespaces in Javascript

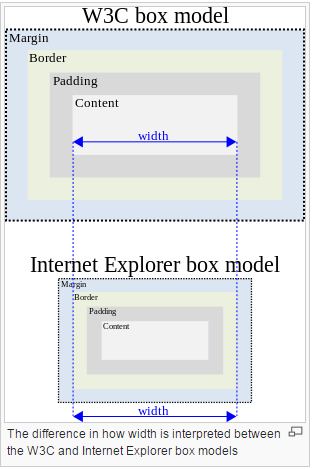
* Unfortunately, the concept of namespaces does not exist in JavaScript.
* Everything in JavaScript is by default **global**
* **single global object**
* Nested namespaces
* Object literal notation
* Immediately-invoked Function Expressions : unnamed functions
* Namespace injection

### Safely creating namespaces : checking the type to be unefined

## Modules: private members through closures

         Box Model Differences in IE / Firefox

When a width or height is explicitly specified for any block-level element, it should determine only the width or height of the visible element, with the padding, borders, and margins applied afterward. [Internet Explorer](http://en.wikipedia.org/wiki/Internet_Explorer) in "[quirks mode](http://en.wikipedia.org/wiki/Quirks_mode)" includes the content, padding and borders within a specified width or height; this results in a narrower or shorter rendering of a box than would result following the standard behavior.



Box model hack" developed by [Tantek Çelik](http://en.wikipedia.org/wiki/Tantek_%C3%87elik" \o "Tantek Çelik): It involves specifying a width declaration for Internet Explorer for Windows, and then overriding it with another width declaration for CSS-compliant browsers. Not successful due to bugs.

 W3C has included a "box-sizing" property in CSS3.

When box-sizing: border-box; is specified for an element, any padding or border of the element is drawn inside the specified width and height, "as commonly implemented by legacy HTML user agents

         CSS Sprites / Basic Performance Issues / Solutions

* + An image sprite is a collection of images put into a single image.

A web page with many images can take a long time to load and generates multiple server requests.

Using image sprites will reduce the number of server requests and save bandwidth.

* + [Combine external JavaScript](https://developers.google.com/speed/docs/best-practices/rtt#CombineExternalJS)
  + [Combine external CSS](https://developers.google.com/speed/docs/best-practices/rtt#CombineExternalCSS)
  + [Combine images using CSS sprites](https://developers.google.com/speed/docs/best-practices/rtt#SpriteImages)
  + [Minify JavaScript](https://developers.google.com/speed/docs/best-practices/payload#MinifyJS)
  + [Minify CSS](https://developers.google.com/speed/docs/best-practices/payload#MinifyCSS)
  + [Minify HTML](https://developers.google.com/speed/docs/best-practices/payload#MinifyHTML)

         UI Debugging tools – like firebug and Ideveloper

         Native JavaScript

         Concept of Block and Inline elements

* + Block level elements normally start (and end) with a new line, when displayed in a browser.
  + Examples: <h1>, <p>, <ul>, <table>, <div>
  + Inline elements are normally displayed without line breaks.
  + Examples: <b>, <td>, <a>, <img>, <span>

         Components of YUI like sortables

         DHTML Treegrid

         Extending the Javascript function and Core DOM API’s

         Concept of Doctypes

**DTD: Document Type Definition**

A Document Type Definition (DTD) defines the legal building blocks of an XML document. It defines the document structure with a list of legal elements and attributes.

A DTD states what [tags](http://www.webopedia.com/TERM/T/tag.html) and [attributes](http://www.webopedia.com/TERM/A/attribute.html) are used to describe content in an [SGML](http://www.webopedia.com/TERM/S/SGML.html), [XML](http://www.webopedia.com/TERM/X/XML.html) or [HTML](http://www.webopedia.com/TERM/H/HTML.html)document, what tags are allowed, and which tags can appear within other tags.

**Document Type Declaration or Doctype:**

A document type declaration, or DOCTYPE, is an instruction that associates a particular [SGML](http://en.wikipedia.org/wiki/SGML) or [XML](http://en.wikipedia.org/wiki/XML) document (for example, a [webpage](http://en.wikipedia.org/wiki/Webpage)) with a [document type definition](http://en.wikipedia.org/wiki/Document_type_definition) (DTD) (for example, the formal definition of a particular version of [HTML](http://en.wikipedia.org/wiki/HTML)).

### HTML 5

<!DOCTYPE html>

### HTML 4.01 Strict

This DTD contains all HTML elements and attributes, but does NOT INCLUDE presentational or deprecated elements (like font). Framesets are not allowed.

### HTML 4.01 Transitional

This DTD contains all HTML elements and attributes, INCLUDING presentational and deprecated elements (like font). Framesets are not allowed.

### HTML 4.01 Frameset

This DTD is equal to HTML 4.01 Transitional, but allows the use of frameset content.

         JSON – Creating a complex JSON object

{

"accounting" : [{ "firstName" : "John", "lastName" : "Doe", "age" : 23 },

{ "firstName" : "Mary", "lastName" : "Smith","age": 32 }

],

"sales" : [{ "firstName" : "Sally", "lastName" : "Green", "age": 27 },

{ "firstName" : "Jim", "lastName" : "Galley", "age": 41 }

]

}

         Experience with libraries like YUI JQuery Prototypes

         Creation of libraries

         How to enable strict mode in JS

         Difference between innerHTML and append();

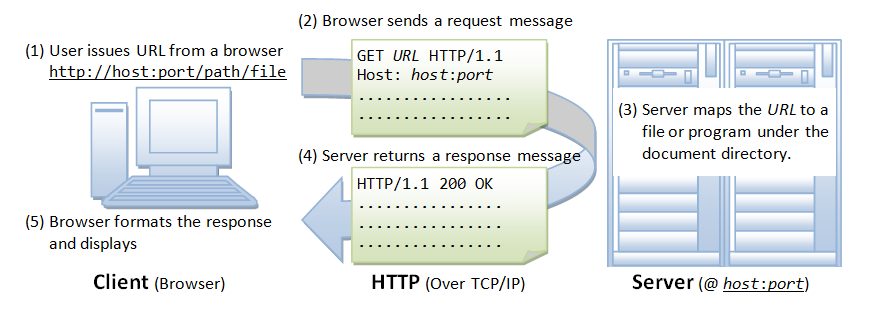
         Number of ways in which object can be created in JS

         Fiddler and Charles for HTTP Requests

Charles is an HTTP proxy / HTTP monitor / Reverse Proxy that enables a developer to view all of the HTTP and SSL / HTTPS traffic between their machine and the Internet. This includes requests, responses and the HTTP headers (which contain the cookies and caching information)

**Web technologies and XML**

HTTP Protocol fundamentals

* The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. This is the foundation for data communication for the World Wide Web (ie. internet) since 1990.
* Major difference between HTTP/1.0 and HTTP/1.1 is that HTTP/1.0 uses a new connection for each request/response exchange where as HTTP/1.1 connection may be used for one or more request/response exchanges.
* HTTP is an TCP/IP based communication protocol, which is used to deliver data
* HTTP specification specifies how clients request data will be constructed and sent to the server, and how servers respond to these requests.
* 
* HTTP Parameters:

1. HTTP Version: HTTP/1.0 or HTTP/1.1
2. URI : http://abc.com:80/~smith/home.html
3. Date and Time format :
4. Character Set: US -ASCII or ISO-8859-1 or ISO-8859-7
5. Content Encoding: Accept-encoding: gzip or Accept-encoding: compress or Accept-encoding: deflate
6. Media Types: **Content-Type** and **Accept**
7. Language Tags : Accept-Language and Content-Language : en, en-US, en-cockney, i-cherokee, x-pig-latin

Servlets basic understanding

JSP’s basic understanding-

Frameworks like Struts, JSF

XML and related technologies

**Object Oriented Principles**

Inheritance /Polymorphism /Encapsulation

Inheritance and Encapsulation are only two concepts apply to OOP in JavaScript.

* **Inheritance** (objects can inherit features from other objects)

Inheritance refers to an object being able to inherit methods and properties from a parent object (a Class in other OOP languages, or a Function in JavaScript).

* **Polymorphism** (objects can share the same interface—how they are accessed and used—while their underlying implementation of the interface may differ)
* **Encapsulation** (each object is responsible for specific tasks).

Encapsulation refers to enclosing all the functionalities of an object within that object, so that the object’s internal workings (its methods and properties) are hidden from the rest of the application. This allows us to abstract or localize specific set of functionalities on objects.

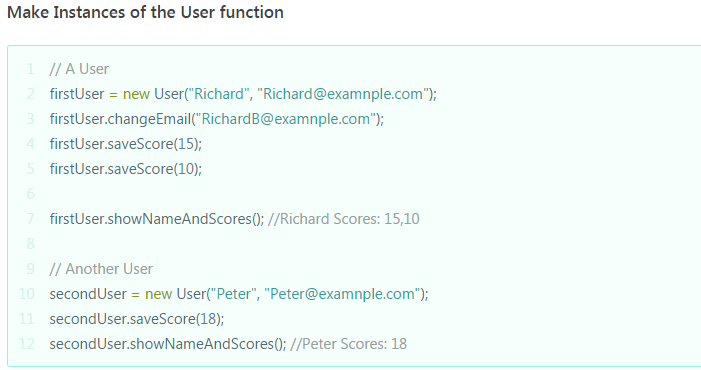
* An instance is an implementation of a Function. In simple terms, it is a copy (or “child”) of a Function or object. For example:

|  |  |
| --- | --- |
|  | // Tree is a constructor function because we will use new keyword to invoke it.​ |
|  | ​function Tree (typeOfTree) {} |
|  | ​ |
|  | ​// bananaTree is an instance of Tree.​ |
|  | ​var bananaTree = new Tree ("banana"); |

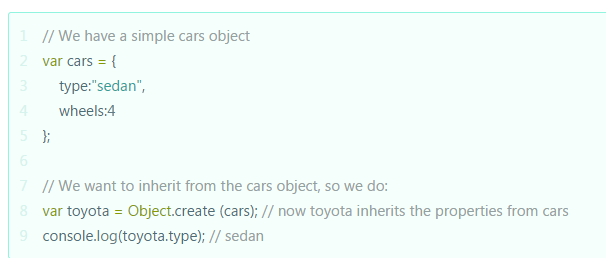
In the preceding example, bananaTree is an object that was created from the Tree constructor function. We say that the bananaTree object is an instance of the Tree object. Tree is both an object and a function, because functions are objects in JavaScript. bananaTree can have its own methods and properties and inherit methods and properties from the Tree object, as we will discuss in detail when we study inheritance below.

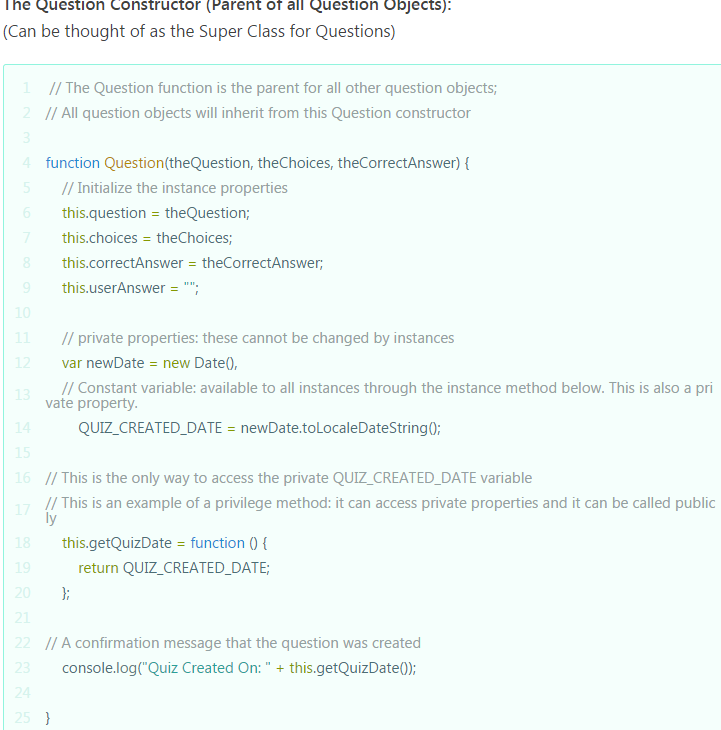
* The two important principles with OOP in JavaScript are Object Creation patterns (Encapsulation) and Code Reuse patterns (Inheritance).
* **Encapsulation in JavaScript**
  + The Best Object Creation Pattern: **Combination Constructor/Prototype Pattern**
  + Whenever you want to create objects with similar functionalities (to use the same methods and properties), you encapsulate the main functionalities in a Function and you use that Function’s constructor to create the objects.
  + **Combination Constructor/Prototype Pattern:**

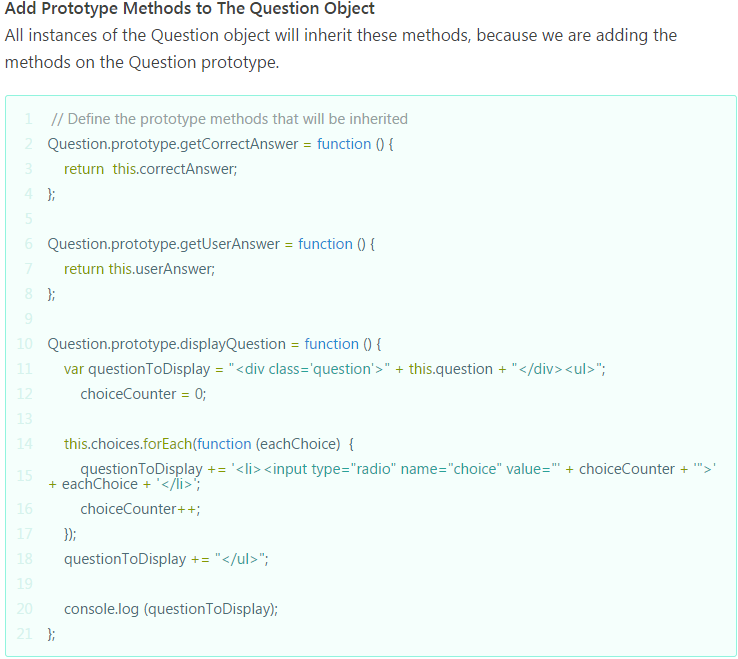


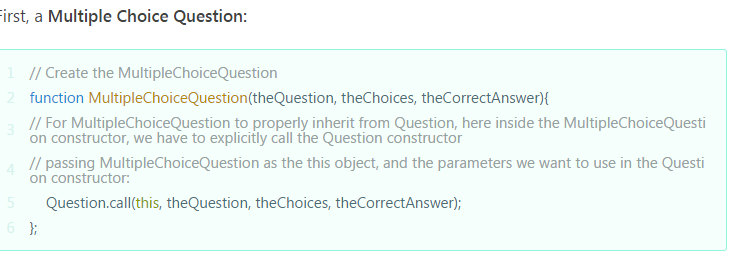


* **Inheritance in JavaScript**
  + The Best Pattern: Parasitic Combination Inheritance

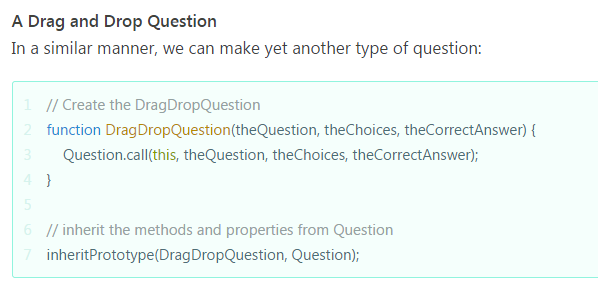


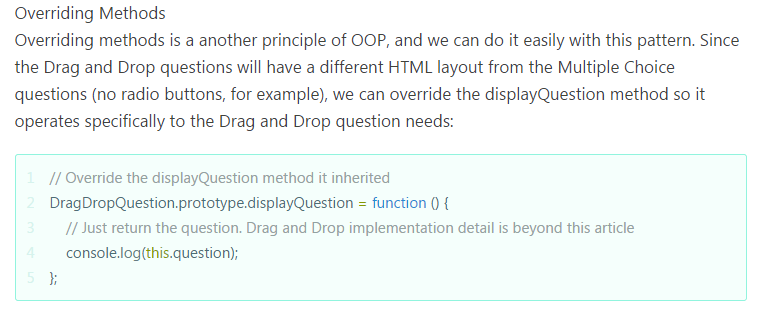












Design patterns like Singleton and Factory

Pass by Reference Vs Pass by Value -

**Java fundamentals**

Language basics like variable types, access modifiers, class hierarchy

Exception Handling

Collections API

Threading basics