HTTP: application level / stateless / used in www / lightweight / συμβατό με πολλά Port / Uri based / MIME

Javascript: DOM object, είναι τα obj. που σχετίζονται με το object document. DOM tree έχει τι περιέχει το σελίδα
- functions are first class objects: i) περνιούνται σαν ορίσματα ii) μπορούν να γίνουν assign iii) -//- return
- Το this σε func είναι global / public, ενώ σε method private. Τα var σε func = private.
- λ-αμδας: map: Επιστρέφει ένα collection τροποποιημένο / forEach: Τροποποιεί τα πεδία ενός collection. / reduce: Επιστρέφει αποτέλεσμα από ένα collection. reduce(...,x) στο result θα έχω + x

object.valueOf() δίνει τι πραγματικά είναι το object
array.splice(2,1) το 2 = index, το 1 = flag = true. Το delete αφήνει τη θέση undefined, το άλλο όχι. Όπως το shift()

func1(){
 func2(){
 }
return func2()
} => var x = func1() x() ← runtime error

func1{
 -//-
}
return fun2
var x = func1 x() ← Σωστό

var = {a:1}; a -> obj.prototype -> null
var b = Obj.create(a) b -> a.proto -> obj.proto -> null
b.a = 1 // Έχω inheritence όπως και στα obj
var d = obj.create(null) d -> null
Άρα d.hasOwnProperty() // undefined
== equality type of objects
=== value equality and type

SERVLETS: Request: Αντικείμενο που περιέχει την πρόθεση του χρήστη, header, parameter
Θετικά: i) Βολικά λόγω Java API / ii) Portable / iii) Αποδοτικά. Κάθε req εξυπηρετείται από ένα thread, όχι process. / Ασφαλή λόγω java έχουμε exceptions και δεν υπάρχουν buffer overflows / διαδεδομένα και φθηνά. Αρνητικά: Υπάρχει μεγάλο overhead / πολλά out.println statements
- Lifecycle: φορτώνεται η servlet class / δημιουργία servlet instance / Καλείται η init() / Καλείται η service() ∀ request / Καλείται η destroy. Ο container καθορίζει τα εξής: πότε ο servlet φορτώθηκε και αρχικοποιήθηκε / άμεση διαχείριση του dispatcher σε άλλο servlet / πότε να σταματήσει ο servlet. ServletConfig: παράγεται από τον container και το παίρνει σαν όρισμα η init(). / Έχουμε ένα μόνο instance ανά servlet => η init() καλείται μόνο μια φορά και όχι για κάθε request. Context Path: Μοναδικό για κάθε application, οι servlet classes στο app έχουν το ίδιο context. / Servlet Path: κάνει map το directory name που σχετίζεται με τον servlet / Path info: extra info, περιέχει τα parameters από το χρήστη.

GET VS POST Request:

GET	POST
μπαίνει στο URL	δεν μπαίνει, μπαίνει στο message body
γίνονται cached	όχι cached
παραμένουν στο history	όχι στο history
γίνονται bookmarks	όχι bookmarks
όχι καλό για sensitive data	καλό για sensitive data
περιορισμοί στο length	όχι length περιορισμούς
Χρήση για retrieve	

Τρόποι Tracking: hidden fields in page / URL-rewritting -> συμπλήρωση πληροφοριών στο URL και αποστολή πίσω / Cookies -> κρατιούνται από τον browser για κάποιο ή για πάντα χρόνο και στο request είναι στο cookies header / Session: Επιλέγει είτε cookies είτε url-re. Τα session δεν μεταφέρονται μέσω network, μόνο τα id τους.

Servlet Scopes: Request scope: local vars σε doGet, doPost. / Only for 1 req: req.setAttribute π.χ. για fw σε άλλο servlet / Only for 1 client: session.setAttribute(key, value) / For anyone: context.setAttribute().

JSP: Είναι server side technology. JSP vs Servlets: Πιο καλή διατήρηση / Χρήση standard HTML Tools. JSP Lifecycle: Μετάφραση JSP σε servlet code / compile του servlet σε byte code / Φόρτωση servlet classes / Δημιουργία server instance / Καλείται η init() req Processing με jspService() / καταστροφή με jspDestroy(). (*) Ενέργειες:
- translation / compile -> page 1st written 1 / page modified 1

- doGet/doPost → σε όλα | Expressions: <%= ... %> γίνεται out.println("");

scriplets: <% code %> Επιτρέπει δηλώσεις (κανονικό κώδικα). Μπαίνει στην jspService

declarations: <%! %> σαν το scriplet με τη διαφορά ότι εδώ το βλέπει όλο το αρχείο, ενώ στο scriplet όχι, είναι τοπικό. | JSP:include VS <%include file="...

JSP:include	<%include file="">
- Σε request time	- Σε translation time
- output of page	- contents of page
- Δεν υπάρχουν response headers που επηρεάζουν το αρχείο	- Υπάρχουν
- ορίζει τα αρχεία που χρησιμοποιεί	- Δεν ισχύει
- from main page #NOT	- Από main page

REST: Χρησιμοποιείται από web apps, χρησιμοποιεί το HTTP για επικοινωνία, URLs to address resources. Τα resources συνδέονται με hyper links. Η αρχιτεκτονική REST περιλαμβάνει: num of reso[urces], μοναδικά IDs (URIs) / ενέργειες με GET, ...

URI Templates: Στην service χρησιμοποιούνται για routing rules / Στο client για init URI από local param[s]

http://cnn.com/story/hawai/fireworks ← REST

http://cnn.com/addComment?name=fireworks ← NOT REST

GET verb accesses a resource, POST creates it. Το HTTP είναι Restfull γιατί:
- Ρήματα (GET, POST) για resource manipulation / - Multiple representations (MIM[E] ...)
- (*) Χρήση Uniform interface γιατί αν GET of http://.../posts/Delete θα χάσω data.

POST is not idempotent PUT is, γιατί POST δημιουργεί πάντα νέο resource ενώ PUT μόνο αν δεν υπάρχει. Άρα POST με ασφάλεια για re-execute.

Διαχείριση resources: PUT, DELETE σε ids από resources / POST: όχι σε ids / GET πα[ντού]

(*) στα δύσκολα αναγράφεται.

/polls	→ GET, POST
/polls/{id}	→ GET, PUT, DEL
/polls/{id}/vote	→ GET, POST
/polls/{id}/vote/{id}	→ GET, PUT

JAX-RS: java API for Restfull web services. πχ. http://ex.com/bookmarks/12

```
@Path("/bookmarks/{bookmark}")
public class BookmarkResource {
  @GET
  public String getBookmark(
   @PathParam("bookmark") String bmID
  }
```