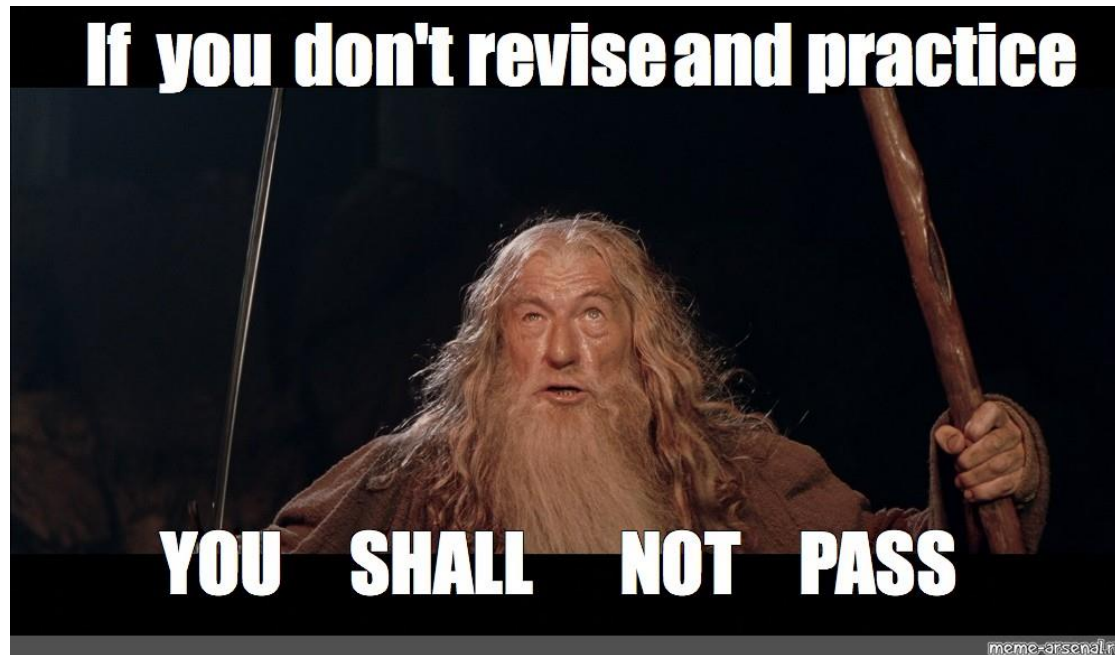


Hy359: Web Programming

Επανάληψη



Instructor: Michalis Mountantonakis
Fall 2022/2023

Διάρθρωση

Part 1

- URIs και HTTP
- Javascript, DOM
- Servers
- Servlets

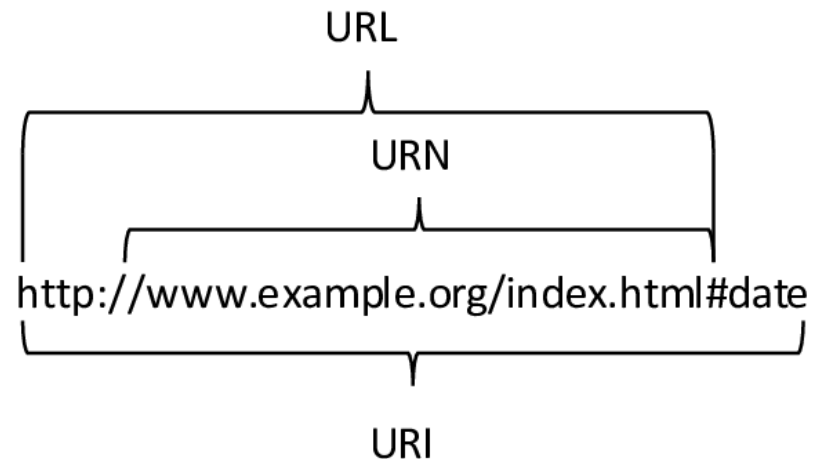
Part 2

- Session Management
- REST
- Άλλα θέματα

URIs kai HTTP

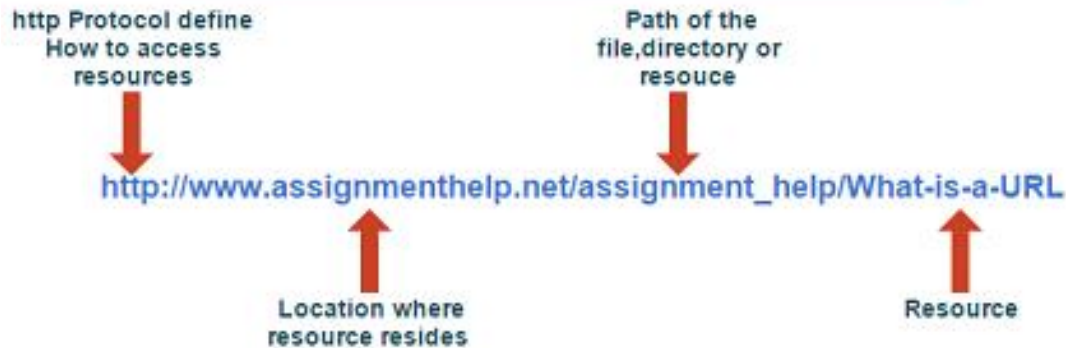
URI, URN, URL, IRI

- ❑ **URI**: Uniform Resource Identifier
 - Ένα String από χαρακτήρες που **προσδιορίζει ένα όνομα ή ένα resource**
 - Σχήματα για τον **καθορισμό της σύνταξης και των πρωτοκόλλων του URI** (syntax and protocols of URI)
- ❑ **URN**: Uniform Resource Name \subset URI
 - **Εξειδικεύει το URI**
 - Το **όνομα ενός προσώπου** (δίνει ταυτότητα σε κάποιο αντικείμενο)
- ❑ **URL**: Uniform Resource Locator \subset URI
 - **Εξειδικεύει το URI**
 - **διεύθυνση δρόμου κάποιου ανθρώπου (μέθοδος εύρεσης αντικειμένου)**
- ❑ **IRI**: International Resource Identifier \supset URI
 - **Επεκτείνει το URI syntax με Unicode**



URI, URN, URL, IRI

Difference between URI, URL and URN



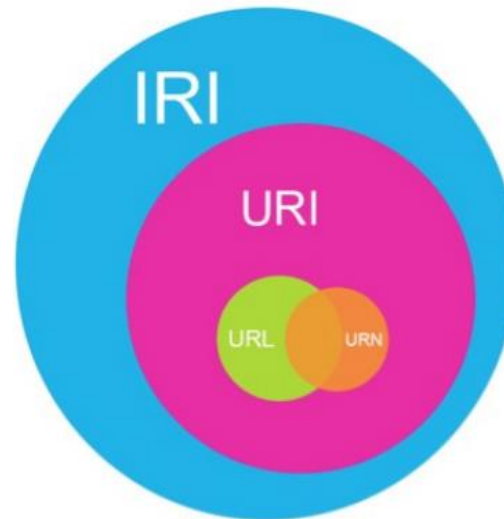
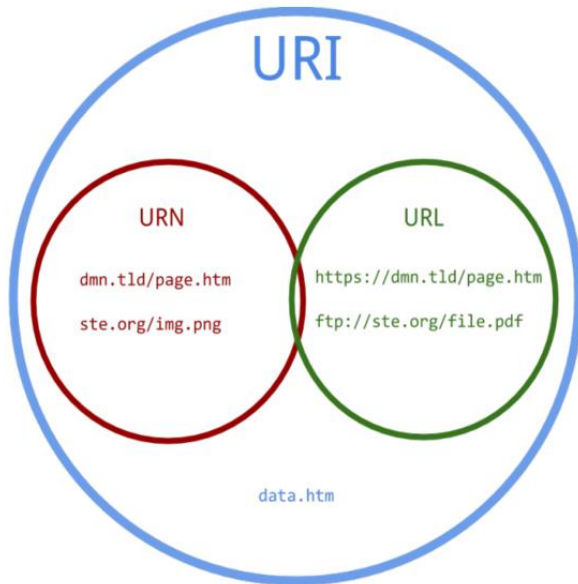
URI: `http://www.assignmenthelp.net/assignment_help/What-is-a-URL`

URL: `http://www.assignmenthelp.net/assignment_help`

URN: `www.assignmenthelp.net/assignment_help/What-is-a-URL`

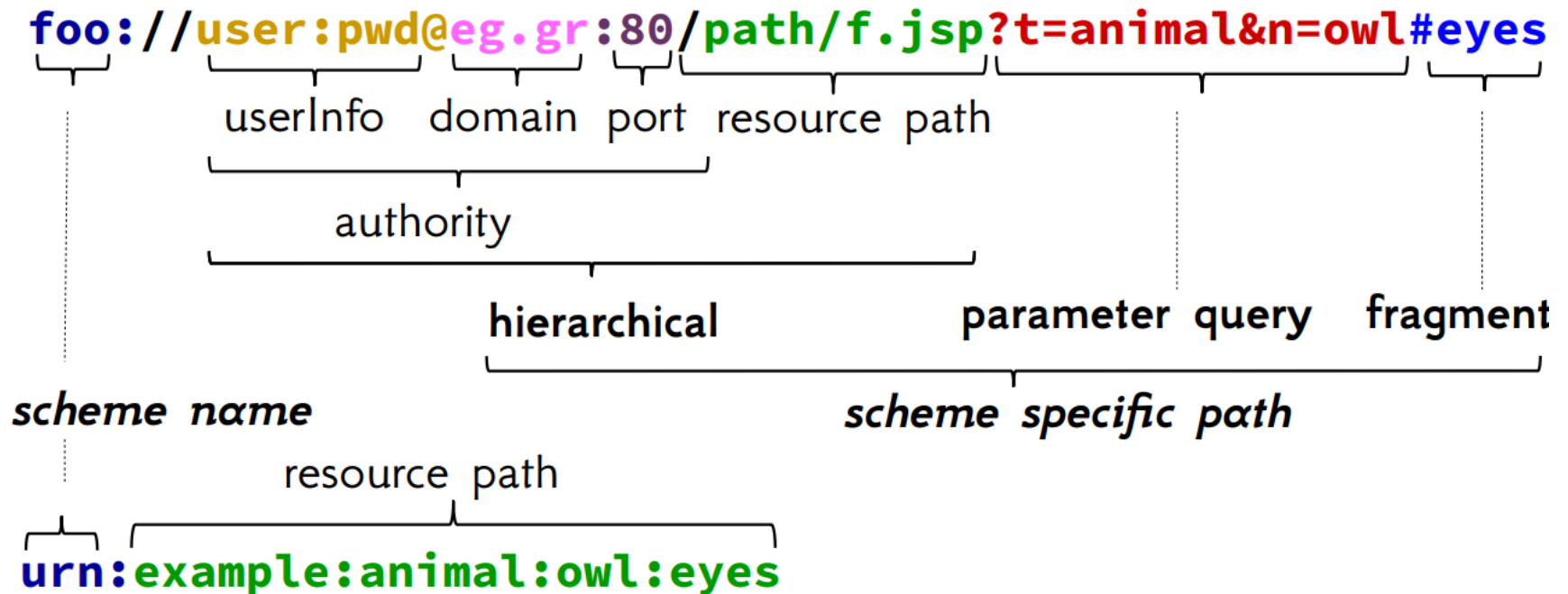
URI, URN, URL, IRI - Κανόνες

- ❑ Κάθε **URL** και κάθε **URN** είναι **URI**
- ❑ Τα **URI** και τα **URL** **ΔΕΝ** είναι εναλλάξιμα
 - Ένα URL είναι πάντα URI, αλλά ένα URI δεν είναι πάντα URL
- ❑ Οι διευθύνσεις URL περιέχουν πάντα έναν μηχανισμό πρόσβασης.
 - http, ftp



URI syntax

URI



URN

URI syntax (1. scheme name)

- Κάθε URI αποτελείται από 4 μέρη
 - **<scheme name>** : <hierarchical part> [? <query>] [# <fragment>]
- 1. Scheme name
 - Το Scheme name αποτελείται από ένα
 - ❖ Γράμμα και ακολουθείται από οποιοδήποτε συνδυασμό γραμμάτων, ψηφίων, '+', '.', ή '-'.
 - ❖ Τερματίζει πάντα με το χαρακτήρα ':'
- Παραδείγματα: ftp:, http:, file:, mailto:

```

foo://example.com:8042/over/there?name=ferret#nose
  \_/_/_/_/_/
  |         |         |         |         |
scheme  authority  path  query  fragment
  |
  \_/_/_/_/_/
urn:example:animal:ferret:nose

```

urn:example:animal:ferret:nose

The diagram illustrates the structure of the URI `urn:example:animal:ferret:nose`. It is divided into five main components: **scheme** (`urn`), **authority** (`example`), **path** (`/`), **query** (`animal`), and **fragment** (`ferret:nose`). The fragment component is further detailed, showing it consists of `ferret` and `nose` separated by a colon.

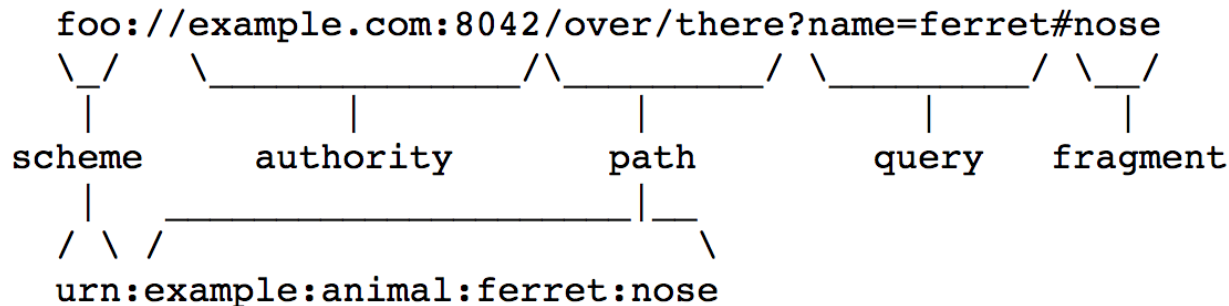
URI syntax (2. hierarchical part)

- Κάθε URI αποτελείται από 4 μέρη

- **<scheme name> : <hierarchical part>** [? <query>] [# <fragment>]

□ Hierarchical part

- Το hierarchical part μας δίνει πληροφορίες ταυτοποίησης, ιεραρχικού χαρακτήρα
- Συνήθως ξεκινά με '//'. Επίσης έχει
 - ❖ Έχει ένα authority part
 - optional user information + '@' (user:passwd@)
 - a hostname, domain or IP address (e.g www.csd.uoc.gr)
 - optional ':' + port number (e.g. :80)
 - ❖ Ένα προαιρετικό path part
 - Sequence of segments, similar to directories, separated by '/'
 - e.g. ~mountant/index.html



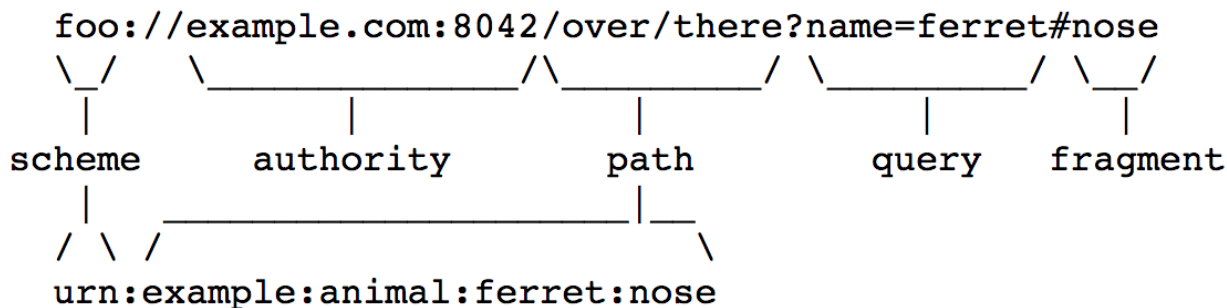
URI syntax (3. Query part)

- Κάθε URI αποτελείται από 4 μέρη

- **<scheme name> : <hierarchical part> [? <query>] [# <fragment>]**

- **Query part**

- Το query part είναι προαιρετικό και το ξεχωρίζουμε **με το question mark (?)**
 - περιέχει **πρόσθετες** και όχι ιεραρχικές πληροφορίες αναγνώρισης
 - Συνήθως οργανώνεται με μία ακολουθία από <key>=<value> ζεύγη που χωρίζονται **με τους χαρακτήρες ; ή &**
 - ❖ <http://www.nba.com/players?team=dallasMavericks&position=Center>
 - ❖ Τρέξτε στο browser το
 - <https://duckduckgo.com/?q=csd>



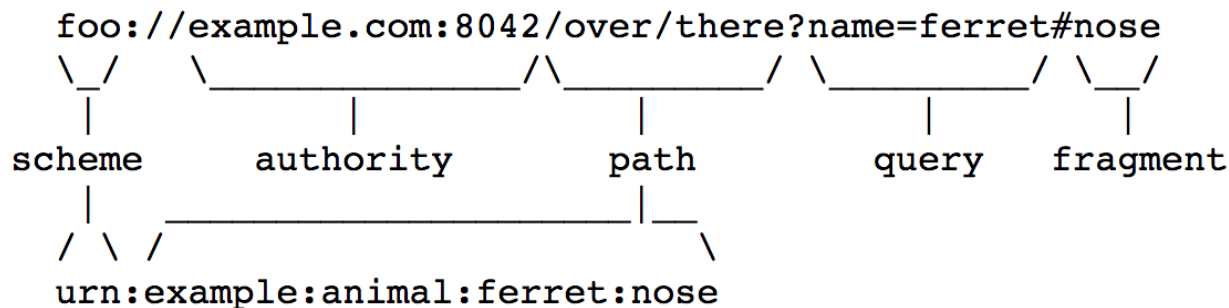
URI syntax (4. Fragment part)

- Κάθε URI αποτελείται από 4 μέρη

- **<scheme name> : <hierarchical part> [? <query>] [# <fragment>]**

- **Fragment part**

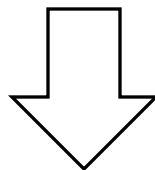
- Το fragment είναι και αυτό προαιρετικό και χωρίζεται από τα άλλα μέρη με το σύμβολο #
- περιέχει πρόσθετες πληροφορίες ταυτοποίησης που παρέχουν κατεύθυνση σε έναν δευτερεύοντα πόρο
 - ❖ Σε ένα συγκεκριμένο section σε ένα άρθρο
 - ❖ Σε ένα συγκεκριμένο id της HTML σελίδας κλπ.



URI Encoding and Normalization

- ❑ Τα URIs μπορούν να χρησιμοποιηθούν με το **ASCII character set**
- ❑ Το URI encoding αντικαθιστά **μη ασφαλής ASCII χαρακτήρες με το '%'** μαζί με 2 **δεκαεξαδικά ψηφία**
 - Πολύ συχνό φαινόμενο
- ❑ **URI normalization**
 - Μετατροπή σε lowercase
 - Capitalize letters σε escape characters
 - Αφαίρεση default port
 - Προσθέτει '/', αφαιρεί dot segments '..', '.', και άλλα

hTtP://www.CSD.uoc.gr:80/~hy359



http%3A%2F%2Fwww.csd.uoc.gr%2F%7Ehy359

Javascript

Τι είναι η Javascript

- ❑ Η JavaScript (JS) είναι μία **διερμηνευμένη γλώσσα** προγραμματισμού για ηλεκτρονικούς υπολογιστές.
 - **Δεν χρειάζεται** να έχει γίνει **compile** προτού τρέξει ο κώδικας
- ❑ Αρχικά αποτέλεσε **μέρος** της **υλοποίησης** των **φυλλομετρητών Ιστού**
 - Για να μπορεί η πλευρά του πελάτη (client-side scripts) να **επικοινωνεί** με τον **χρήστη**, να **ανταλλάσσουν δεδομένα ασύγχρονα** και να αλλάζουν **δυναμικά** το **περιεχόμενο** του **εγγράφου** που εμφανίζεται
- ❑ Είναι δυναμική (**dynamic-type checking**)
 - **Επαληθεύει** τον κώδικα σε **πραγματικό χρόνο** την ώρα της εκτέλεσης (type safety at runtime)
 - ❖ Μπορεί να οδηγήσει σε αρκετά **runtime errors**
 - συντακτικά λάθη, λάθη σε τύπους...
 - Βασίζεται στα πρωτότυπα (**prototype-based**)
 - ❖ **Δεν έχει κλάσεις**
 - Με **ασθενείς τύπους** και έχει **συναρτήσεις** ως αντικείμενα πρώτης τάξης
- ❑ Επίσης είναι μία **multi-paradigm** γλώσσα
 - Imperative & object-oriented & Functional
- ❑ Είναι **Garbage collected**
- ❑ Μπορεί να τρέξει σε **διαφορετικές πλατφόρμες**

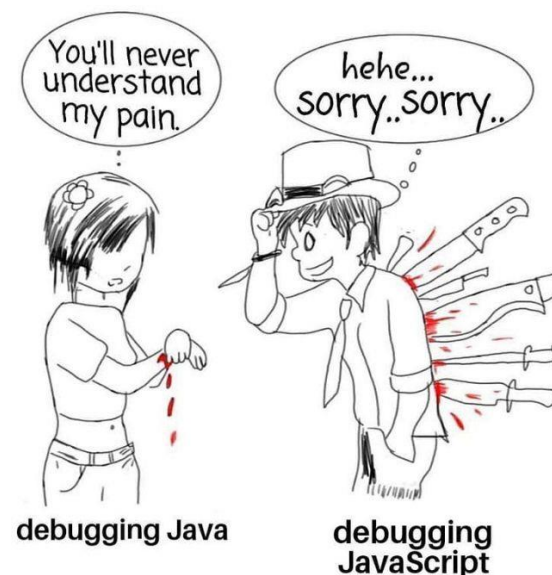
Javascript vs Java?

- Η **JavaScript** αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη **Java**, αλλά **γενικά οι δύο αυτές γλώσσες δε σχετίζονται** και έχουν πολύ διαφορετική σημασιολογία.

Javascript	Java
Διερμηνευμένη (interpreted)	Compiled
Δυναμική	Στατική
Βασίζεται στα πρωτότυπα	Βασίζεται σε κλάσεις
Function-based Scoping	Block-based Scoping
Τρέχει στο Browser, μαζί με την HTML	Χρειάζεται JVM
Ο Κώδικας είναι προσβάσιμος	Ο Κώδικας είναι μη προσβάσιμος



ILLUSTRATED BY SEQUE TECHNOLOGIES



Variables

- ❑ Τρεις τρόποι να ορίσει κάποιος μεταβλητές
 - **var**
 - ❖ Ο πιο **συνήθης τρόπος**
 - ❖ Μπορεί και να παραληφθεί (καλύτερα όχι όμως)
 - **let**
 - ❖ Χρησιμοποιούνται **μέσα σε block**
 - **const**
 - ❖ Δεν αλλάζει η τιμή τους
- ❑ Προσοχή!!! Οι μεταβλητές είναι **Case sensitive**
 - Ξεκινούν με {a,...,z,A,...,Z,_,_\$}
- ❑ scope
 - **local**
 - ❖ Αν χρησιμοποιηθεί var σε function, δεν είναι ορατή έξω από τη function
 - **global**
 - ❖ Στις άλλες περιπτώσεις
 - ❖ Αν δε χρησιμοποιηθεί το var
 - **block**
 - ❖ Χρήση let

```
var str; // add also value

var strName = 'JavaScript'; // global

test = 'JavaScript';

var a,b,c;

var name='Mike', age=29;

const con = 12;
con = 13; // error

function myFun() {
    var a; // local
    g = 'global'; // global
}

myFun(); // have to invoke
console.log(g); // global
{
    let block = 'block'
}
console.log(block); // error
```


Variables (var vs let vs const)

keyword	const	let	var
global scope	NO	NO	YES
function scope	YES	YES	YES
block scope	YES	YES	NO
can be reassigned	NO	YES	YES

Types

- Το **typeof** επιστρέφει μία **string αναπαράσταση** που μας λέει τον **τύπο** της **μεταβλητής**

Τύπος	Τιμές	typeof
string	"hy359", "csd"	'string'
boolean	True, false	'boolean'
object	document, form, user-defined	'object'
function	Any built-in or user defined function, e.g. isNaN()	'function'
number	1, -10, 0.3, ...	'number'
Null	Null	'object'
undefined	undefined	'undefined'
Array	[1,2,'ciao']	'object'

Operators

□ Αριθμητικοί

- + - * / % + --

□ Assignment

- = += -= *= /= %=

□ Comparison

- == != > >= <= <
 - ❖ these operators do type coercion
- === !==
 - ❖ no type coercion with them (safer!)

□ Logical Operators

- && || !

□ String

- + concatenate

```
var str = '8';  
var num = 8;
```

```
str == num  
// type coercion  
// returns true
```

```
str === num  
// no type coercion  
// return false
```

expr1 && expr2

Returns expr1 if it evaluates to false,
otherwise returns expr2

expr1 || expr2

Returns expr1 if it evaluates to true,
otherwise returns expr2

Αυτόματη μετατροπή τύπου (coersion)

- ❑ Type coercion converted operand(s) to an "equivalent" value(s)
 - when the operands of an operator are different types
 - when an operator is applied to invalid types, like * in strings

- ❑ $5 + \text{null} = 5$
 - null is converted to 0
- ❑ $'5' + \text{null} = '5\text{null}'$
 - null is converted to 'null'
- ❑ $'5' + 2 = '52'$
 - 2 is converted to '2'
- ❑ $'5' - 2 = 3$
 - '5' is converted to 5
- ❑ $'5' * '2' = 10$
 - '5' and '2' are converted to 5 and 2
- ❑ $1 + 2 + '3' = '33'$
 - left associativity

Εισαγωγή στα Objects

- ❑ Properties/Attributes
- ❑ Ένα property μπορεί να προστεθεί σε ένα αντικείμενο μέσω μιας απλής ανάθεσης
 - `a.x = 5;` // property x did not exist
 - Αν δεν έχει ανατεθεί κάποια τιμή, τότε θεωρούνται undefined
 - Μπορούμε να έχουμε πρόσβαση με 2 τρόπους
 - ❖ `obj.member1` ή `obj['member1']`
- ❑ Properties (μόνο!) μπορούν να διαγραφούν μέσω του operator delete
 - `delete a.x;`
 - Μπορούν να οδηγήσουν σε μη αναμενόμενα αποτελέσματα (λόγω κληρονομικότητας inheritance)

Για να διατρέξουμε όλα τα
στοιχεία του αντικειμένου

```
for (var name in object) {  
    // name is the key of the current member  
    object[name]  
}
```

Δημιουργία Objects

- ❑ 4 διαφορετικοί τρόποι

- Με τη χρήση του operator **new**
- Μέσω templates
- Μέσω Object literals
- Με τη χρήση της `Object.create()`; // ECMAScript 5 (prototypical inheritance)

Δημιουργία Objects- Χρήση new Operator

Χρήση new Operator

```
// Initially no properties/methods
var john = new Object();
john.firstName = 'John';
john.age=32;
john.tellYourAge=function(){
  alert('My age is'+this.age); }
// An object can have another object as property
john.daughter={};
john.daughter.age=2;
console.log(john.daughter.age);
```

Δημιουργία Objects - Χρήση templates

- ❑ Τίποτα παραπάνω από μία κανονική συνάρτηση
- ❑ Το πρώτο γράμμα πρέπει να είναι κεφαλαίο
- ❑ Το `this` αναφέρεται στο νέο object που θα δημιουργηθεί
- ❑ **To new** δημιουργεί ένα κενό object και κάνει assign το `this` σε αυτό

```
function Person(fname, lname, age){  
    this.fname=fname;  
    this.lname=lname;  
    this.age=age;  
    this.tellYourAge=function(){  
        alert('My age is ' + this.age);  
    }  
    Person.anotherProp='Will not appear!';  
    var father=new Person('John', 'Doe', 58);  
    var mother=new Person('Saly', 'Raly', 53);  
    console.log(father.tellYourAge());  
}
```


Δημιουργία Objects - Χρήση templates

- ❑ Το αντικείμενο μπορεί να φτιαχτεί και μέσα στη συνάρτηση

```
function Violin(violinName, makerName, where, year, grade) {  
    var it = {}; // {} creates an empty object  
    it.violinName = violinName;  
    it.makerName = makerName;  
    it.where = where;  
    it.year = year;  
    it.grade = grade;  
    return it;  
}  
  
var ilCannone = Violin('Il Cannone Guarnerius', 'Bartolomeo Giuseppe Antonio Guarneri', 'Cremona', 1743, 'MasterPiece');  
  
console.log(ilCannone);
```

Δημιουργία Objects – Object Literals

- ❑ Τα object literals δημιουργούνται μέσα σε άγκιστρα {}
- ❑ Εκφράσεις τύπου όνομα-τιμή (name-value)
 - Name: σαν μία μεταβλητή
 - Value: μπορεί να είναι οποιαδήποτε έκφραση (expression)
 - **Άνω κάτω τελεία** : διαχωρίζει names και values
 - **Κόμμα** , διαχωρίζει name-value pairs
- ❑ Τα object literals μπορούν να εμφανιστούν οπουδήποτε μπορεί να εμφανιστεί μια τιμή

```
var ilCannone = {  
  violinName : 'Il Cannone Guarnerius',  
  makerName : 'Bartolomeo Giuseppe Antonio Guarneri',  
  where :  
    { location: 'Cremona', country: 'Italy' },  
  year : 1743,  
  grade : 'MasterPiece',  
  play : function(score) {  
    alert(score);  
  }  
}  
ilCannone.play('5thSymphony');
```

DOM document

explicit dom tree references

- ❑ Κάποιος μπορεί να έχει πρόσβαση σε κάποιο στοιχείο με πολλούς τρόπους
 - **getElementById()** μέθοδος του document object
 - **getElementsByTagName()** μέθοδος του document object
 - **getElementsByName()** μέθοδος του document object
 - **getElementsByClassName()** μέθοδος του document object

- ❑ traversing the tree using methods such as:
 - getChildNodes()
 - getNextSibling()
 - K.α.

Πως αλλάζω το περιεχόμενο της HTML?

- ❑ Αλλαγή Κειμένου – κώδικα (πχ σε div)
 - Χρήση innerHTML
 - ❖ `document.getElementById("message").innerHTML="αα";`
- ❑ Πως θα πάρουμε το κείμενο
 - Χρήση innerHTML
 - ❖ `var value=document.getElementById("message").innerHTML;`

getElementById() Example

```
<html>
<head>
<script>
var times=0;
function myFunction() {
  var x = document.getElementById("demo");
  x.style.color = "red";
  document.getElementById("message").innerHTML+="<br> Pushed "+times++;
}
</script>
</head>
<body>
<p id="demo"> Click the button to change the color of this paragraph and the div
message</p>
<button onclick="myFunction()">Try it</button>
<div id="message"></div>
</body>
</html>
```

getElementById()

getElementsByTagName() Example

```
<!DOCTYPE html>
<html>
<head>
  <script>
function myFunction() {
  var x = document.getElementsByTagName("p");
  var i;
  for (i = 0; i < x.length; i++) {
    x[i].style.backgroundColor = "yellow";
  }
}
</script>
</head>
<body>
<p>This is a p element</p>
<p>This is also a p element.</p>
<p>This is also a p element -
Click the button to change the background color of all p elements in this document.</p>
<button onclick="myFunction()">Try it</button>
</body>
</html>
```

getElementsByTagName()

getElementsByTagName() Example

```
<!DOCTYPE html>
<html>
<body>
Cats: <input name="animal" type="checkbox" value="Cats">
Dogs: <input name="animal" type="checkbox" value="Dogs">
<p>Click the button to check all checkboxes that have a name attribute with the value "animal".</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction() {
  var x = document.getElementsByTagName("animal");
  var i;
  for (i = 0; i < x.length; i++) {
    if (x[i].type == "checkbox") {
      x[i].checked = true;
    }
  }
}
</script>
</body>
</html>
```

getElementsByTagName()

getElementsByClassName() Example

```
<!DOCTYPE html>
<html>
<body>
<div class="example">
A div element with class="example"
</div>
<div class="example">
Another div element with class="example"
</div>
<p class="example">A p element with class="example"</p>
<p>Click the button to find out how many elements with class "example" there are in this document
.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
  var x = document.getElementsByClassName("example");
  document.getElementById("demo").innerHTML = x.length;
}
</script>
</body>
</html>
```

getElementsByClassName()



Functions (Τα βασικά)

❑ Παραδείγματα invocation (επίκλησης)

- Μέσω ενός event
- Μέσω Javascript
- Μέσω αυτόματης επίκλησης (ανώνυμες συναρτήσεις)

```
function myFn(x, y) {  
    return x * y;  
} // No ; here since it is not executed  
  
myFn(1,2);  
  
(function() { // Can use ! in front instead of (function(){...})  
    alert('Self Invoked function'); })();  
// Self-invoked anonymous function
```

Functions Invocation

- ❑ **Υπάρχουν τέσσερις τρόποι να καλέσετε μία συνάρτηση:**
 - Function form
 - ❖ `functionObject(arguments)`
 - Method form
 - ❖ `thisObject.methodName(arguments)`
 - ❖ `thisObject["methodName"](arguments)`
 - Constructor form
 - ❖ `new functionObject(arguments)`
 - Apply and Call form
 - ❖ `functionObject.apply(thisObject,[arguments])`
 - ❖ `functionObject.call(thisObject,arg1, arg2,...)`

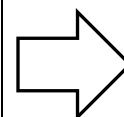
Functions Invocation και το keyword this

❑ 1. Function form

- fn(arguments);
- Όταν καλείται μία συνάρτηση μέσω function form, το keyword this δείχνει στο global object

In HTML the value of **this**, in a global function, is the window object.

```
let x = myFunction();  
function myFunction() {  
  return this;  
}  
document.getElementById("demo").innerHTML = x;
```



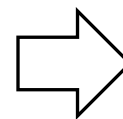
[object Window]

Functions Invocation και το keyword this

❑ 2. Method form

- object.method(arguments);
- object[method](arguments);
 - ❖ myObject['fullName']();
- Όταν μία συνάρτηση καλείται μέσω method form, **το keyword this αντιστοιχεί το αντικείμενο που περιέχει τη συνάρτηση**
- Επιτρέπει στις μεθόδους να έχουν reference στο αντικείμενο

```
var myObject = {  
    firstName: "John",  
    lastName: "Doe",  
    fullName: function() {  
        return this.firstName + " " + this.lastName;  
    }  
}  
document.getElementById("demo").innerHTML =  
myObject.fullName();
```



[John Doe]

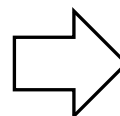
Functions Invocation και το keyword this

❑ 3. Constructor form

- `new constructor(arguments);`
- Όταν μία function καλείται μέσω του new operator, **ένα νέο αντικείμενο δημιουργείται και γίνεται ανάθεση στο this.**
- **Αν δεν υπάρχει κάποιο return value, τότε θα επιστραφεί το this.**
- It looks like you create a new function, but since JavaScript functions are objects you actually create a new object.

```
function myFunction(arg1, arg2) {  
  this.firstName = arg1;  
  this.lastName = arg2;  
}
```

```
const myObj = new myFunction("John","Doe");  
document.getElementById("demo").innerHTML =  
myObj.firstName;
```



[John]

Functions Invocation και το keyword this

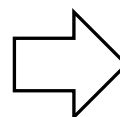
❑ 4. apply and call form

- `fn.apply(object, [arguments]);`
- `fn.call(object, arg1, arg2, ...);`
- ❑ Η `apply` μέθοδος καλεί μία συνάρτηση με το `this` σαν τιμή και τα `arguments` παρέχονται ως πίνακας
- ❑ Το ίδιο για το `call`, αλλά εκεί δίνουμε ξεχωριστά κάθε `argument`

```
const person = {  
  fullName: function() {  
    return this.firstName + " " + this.lastName;  
  }  
}
```

```
const person1 = {  
  firstName: "Mary",  
  lastName: "Doe"  
}
```

```
// This will return "Mary Doe":  
person.fullName.apply(person1);
```

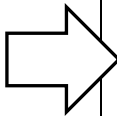


Mary Doe

parameters

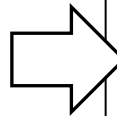
- ❑ Τα αντικείμενα μπορούν να **περνάνε ως παράμετροι** στα **functions** και μπορούν να **επιστραφούν από functions**
 - Τα objects και οι πίνακες **περνάνε/επιστρέφονται by reference**
 - Οι μεταβλητές **περνάνε by value**
- ❑ Τα **functions** είναι **επίσης objects**, άρα **το ίδιο ισχύει για τα functions**
- ❑ Οι παράμετροι που λείπουν (missing parameters) παίρνουν την τιμή **undefined**
- ❑ **Μπορούν να δοθούν και default τιμές!!**
 - `function lala(a, b=10) {}`
 - ❖ Σημαίνει ότι η default τιμή του b είναι το 10

```
function myFunction(x, y = 2) {  
  return x * y;  
}  
document.getElementById("demo")  
  .innerHTML = myFunction(4);
```



8

```
function myFunction(x, y = 2) {  
  return x * y;  
}  
document.getElementById("demo")  
  .innerHTML = myFunction(4,5);
```



20

Pass By Reference

- ❑ Στο Pass by Reference, η συνάρτηση καλείται περνώντας απευθείας την αναφορά της μεταβλητής ως όρισμα. Η αλλαγή του ορίσματος μέσα στη συνάρτηση επηρεάζει τη μεταβλητή που μεταφέρεται έξω από τη συνάρτηση.
 - **Στα αντικείμενα και τους πίνακες Javascript έχουμε pass by reference**

```
function myFunction(varObj) {  
    console.log("Inside Call by Reference Method");  
    varObj.a = 100;  
    console.log(varObj);  
}  
let varObj = {a:1};  
console.log("Before Call function MyFunction");  
console.log(varObj);  
myFunction(varObj);  
console.log("After Call function MyFunction");  
console.log(varObj);  
output will be :  
-----  
Before Call function MyFunction  
{a: 1}  
Inside Call function MyFunction  
{a: 100}  
After Call function MyFunction  
{a: 100}
```

Pass By Value

- ❑ Στο Pass by Value, η συνάρτηση καλείται περνώντας απευθείας την τιμή της μεταβλητής ως όρισμα.
 - Η αλλαγή του ορίσματος μέσα στη συνάρτηση δεν επηρεάζει τη μεταβλητή που μεταφέρεται από έξω από τη συνάρτηση.

```
function myFunction(varOne, varTwo) {  
  console.log("Inside Call by Value Method");  
  varOne = 100;  
  varTwo = 200;  
  console.log("varOne =" + varOne + "varTwo ="  
+varTwo);  
}  
let varOne = 10;  
let varTwo = 20;  
console.log("Before Call function MyFunction ");  
console.log("varOne =" + varOne + "varTwo =" +varTwo);  
myFunction(varOne, varTwo)  
console.log("After Call function MyFunction ");  
console.log("varOne =" + varOne + "varTwo =" +varTwo);
```

Output will be :

Before Call function MyFunction
varOne =10 varTwo =20

Inside Call function MyFunction
varOne =100 varTwo =200

After Call function MyFunction
varOne =10 varTwo =20

arguments

- Όταν μία function ενεργοποιείται, εκτός από τις παραμέτρους, έχει και μία extra ειδική παράμετρο με όνομα **arguments**!
 - Περιλαμβάνει όλα τα arguments της κλήσης
 - Είναι ένα αντικείμενο σαν πίνακας
 - Passed by Value (όταν είναι μεταβλητές)
 - ❖ Προσοχή σε αντικείμενα
 - arguments.length ο αριθμός των arguments

```
function sum(param1, param2, param3) {  
    var i, n = arguments.length, total = 0;  
    for (i = 0; i < n; i += 1) {  
        total += arguments[i];  
    }  
    return total;  
}
```

Sum(1,2,3)=6

Sum(1,2)=3

sum(1,2,5,6)=14

Functions as function parameters (Callbacks)

- Απλά περάστε το όνομα της συνάρτησης ως παράμετρο
 - Σημειώστε ότι δεν υπάρχουν παρενθέσεις () μετά το όνομα της συνάρτησης που δίνεται ως παράμετρος

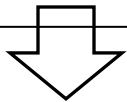
```
function test(id) {  
  alert('hello hy359 class' + id);  
};  
function addContact(id, callback) {  
  callback(id);  
  // You can also pass arguments if  
  // you need to  
};  
addContact(2021, test);
```

```
function myDisplayer(some) {  
  document.getElementById("demo").i  
  nnerHTML = some;  
}  
  
function myCalculator(num1, num2,  
  myCallback) {  
  let sum = num1 + num2;  
  myCallback(sum);  
}  
  
myCalculator(5, 5, myDisplayer);
```

return functions

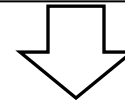
- ❑ Οι functions μπορούν να επιστραφούν σαν return values άλλων functions
 - Χρήσιμα και για τα closures

```
function a() {  
    alert('a');  
    function b() {  
        alert('b');  
        // returns undefined  
    };  
    return b(); // undefined  
};  
var s = a();  
alert('another call');  
s();
```



```
a  
b  
another call  
!runtime error
```

```
function a() {  
    alert('a');  
    function b() {  
        alert('b');  
        // returns undefined  
    };  
    return b; // function  
};  
var s = a();  
alert('another call');  
s();
```



```
a  
another call  
b
```

Παραδείγματα

```
function square(x) {  
    return x * x;  
}  
function triple(x) {  
    return x * 3;  
}  
function doOperation(f,x) {  
    return f(x);  
}  
doOperation(square, 5); // 25  
doOperation(triple, 10); // 30  
//pinakas apo functions  
  
var functions = [square, triple];  
functions[0](10); // 100  
functions[1](20); // 60
```

```
function A(x){  
    return x * (x - 1);  
}  
function B(f,x) {  
    return (f(x));  
}  
function C(f, n) {  
    return(function(x) {  
        return f(x) + n;  
    });  
}  
A(4); //12  
B(A, 4); //12  
var param = 4;  
var test = C(A, param);  
test(1); //4  
param = 3;  
test(1); //4  
test = C(A, param); //
```

```
// function(x)  
{return A(x) +  
param;}
```

Scope Μεταβλητών

□ Προσοχή

- Το scope αλλάζει μέσα στις function, αλλά πρέπει να οριστεί ξανά η μεταβλητή μέσω της χρήσης `const` `let` ή `var`
 - ❖ Αλλιώς είναι `global`
- Οι μεταβλητές που δίνονται ως παράμετροι, περνούν `by value`
 - ❖ Αν αλλάξει η τιμή τους, δεν επηρεάζει τις μεταβλητές έξω από τη function
- Τα αντικείμενα που δίνονται ως παράμετροι, περνούν `by reference`
- Οι function μπορούν να έχουν `default` τιμές, αλλά αυτές ισχύουν μόνο αν δε δοθεί η αντίστοιχη παράμετρος

keyword	const	let	var
global scope	NO	NO	YES
function scope	YES	YES	YES
block scope	YES	YES	NO
can be reassigned	NO	YES	YES

Timing events

setInterval

- ❑ The `setInterval()` method calls a function or evaluates an expression at specified intervals (in milliseconds).
- ❑ The `setInterval()` method will continue calling the function until `clearInterval()` is called, or the window is closed.
- ❑ The ID value returned by `setInterval()` is used as the parameter for the `clearInterval()` method.
- ❑ **Tip:** 1000 ms = 1 second.
- ❑ **Tip:** To execute a function only once, after a specified number of milliseconds, use the `setTimeout()` method.

setTimeout

- ❑ The `setTimeout()` method calls a function or evaluates an expression after a specified number of milliseconds.
- ❑ **Tip:** 1000 ms = 1 second.
- ❑ **Tip:** The function is only executed once. If you need to repeat execution, use the [`setInterval\(\)`](#) method.
- ❑ **Tip:** Use the [`clearTimeout\(\)`](#) method to prevent the function from running.

setTimeout

```
<!DOCTYPE html>
<html>
<body>

<p>Click the first button alert "Hello" after waiting 3 seconds.</p>
<p>Click the second button to prevent the first function to execute.
(You must click it before the 3 seconds are up.)</p>

<button onclick="myFunction()">Try it</button>
<button onclick="myStopFunction()">Stop the alert</button>

<script>
var myVar;

function myFunction() {
  myVar = setTimeout(function(){ alert("Hello") }, 3000);
}

function myStopFunction() {
  clearTimeout(myVar);
}
</script>

</body>
</html>
```

Servers

Instructions to Server - HTTP requests

- ❑ **Safe methods** – δεν τροποποιούν το resource στο server side.
- ❑ **Όλες οι safe HTTP methods είναι idempotent** αλλά οι PUT και DELETE είναι idempotent αλλά όχι safe
- ❑ **Idempotency** σημαίνει ότι πολλαπλά requests μπορούν να έχουν το ίδιο αποτέλεσμα
 - Άρα δεν έχει σημασία αν το request στάλθηκε μία η πολλές φορές
 - Οι ακόλουθες HTTP methods είναι idempotent: GET, HEAD, OPTIONS, TRACE, PUT and DELETE.
- ❑ **PUT vs POST:**
 - Η POST φτιάχνει ένα νέο αντικείμενο
 - Η PUT συνήθως αλλάζει ένα ήδη δημιουργημένο αντικείμενο
- ❑ **PUT & DELETE:** used in APIs (e.g. REST)

HTTP Method	Safe	Idempotent
OPTIONS	yes	yes
GET	yes	yes
HEAD	yes	yes
PUT	no	yes
POST	no	no
DELETE	no	yes
PATCH	no	no

HTTP Request: Basic methods

- ❑ **GET:** Ανακτά τους πόρους από τον server και τους επιστρέφει στον client
 - Η μέθοδος GET ζητά από το διακομιστή να στείλει τη σελίδα. Η σελίδα κωδικοποιείται κατάλληλα σε μορφή MIME. Η πιο συνηθής μορφή της μεθόδου GET είναι η εξής:
 - **GET <URL> HTTP/1.1: Βήματα**
 - ❖ 1. Αποστολή του request
 - Να διαβάσω μία είδηση
 - ❖ 2. Ανάκτηση του κατάλληλο πόρου από τον server
 - ❖ Χαρακτηριστικά
 - Το query part δεν μπορεί να ξεπερνά τους 1024 χαρακτήρες
 - Μπορεί να γίνει cached & bookmarked
 - ❖ Παράδειγμα
 - GET www.csd.uoc.gr HTTP/1.0
- ❑ **HEAD:** Σαν την GET αλλά επιστρέφει μετά-πληροφορίες για τον πόρο
 - όχι τον ίδιο τον πόρο

HTTP Request: Basic methods

- ❑ **POST:** Στέλνει δεδομένα από τον client στον server
 - Ενημερώνει τα περιεχόμενα του server (π.χ. δημιουργία νέου πόρου)
 - ❖ Φόρμα εγγραφής
 - Το Query part ενσωματώνεται στο body content (όχι στο URL)
 - ❖ Δεν υπάρχει περιορισμός 1024 χαρακτήρων
 - ❖ Ο χρήστης δε το βλέπει
- ❑ **PUT, DELETE:** Προσθέτει/Ανανεώνει/Αφαιρεί ένα πόρο σε ένα συγκεκριμένο URI
- ❑ **Άλλα:** TRACE, OPTIONS, CONNECT, PATCH

Παράδειγμα

- ❑ **Εγγραφή Χρήστη**
 - **POST Request**
- ❑ **Ανάκτηση στοιχείων χρήστη**
 - **GET Request**
- ❑ **Ανανέωση πεδίου χρήστη**
 - **PUT Request**
- ❑ **Διαγραφή Χρήστη**
 - **DELETE Request**

HTTP Headers – Some Examples

❑ Request

- **User Agent:** Info about the client (e.g. browser version)
- **Accept:** What MIME-types are accepted by the client (e.g. image/jpeg)
- **Accept-Language:** What language to accept (e.g. en-us)
- **Accept-Charset:** What encoding to accept (e.g. ISO-8859-1)
- **Connection:** What to do with the connection after the request (e.g. keep-alive)
- **Referer:** allow the client to tell the server the URL of the document that triggered this request (trace clients)
- **Authorization:** Access control

❑ Response

- **Server:** Information about the server (e.g. Apache tomcat)
- **Content-Type:** Specifies a MIME type to describe how the document should be interpreted –
- **Location:** If the document has moved provide the new location
- **WWW-authenticate:** Access control
- **Content-Length:** Determine when all data has been received
- **Last-Modified:** When was this document last modified
- **Cache-control:** Age of cache (e.g. max-age=120)
- **Etag:** A hash used for caching

Παράδειγμα HTTP Request- Response

```
GET /doc/test.html HTTP/1.1
```

```
Host: www.test101.com
```

```
Accept: image/gif, image/jpeg, */*
```

```
Accept-Language: en-us
```

```
Accept-Encoding: gzip, deflate
```

```
User-Agent: Mozilla/4.0
```

```
Content-Length: 35
```

```
bookId=12345&author=Tan+Ah+Teck
```

Request Line

Request Headers

Request
Message
Header

A blank line separates header & body

Request Message Body

Request

```
HTTP/1.1 200 OK
```

```
Date: Sun, 08 Feb xxxx 01:11:12 GMT
```

```
Server: Apache/1.3.29 (Win32)
```

```
Last-Modified: Sat, 07 Feb xxxx
```

```
ETag: "0-23-4024c3a5"
```

```
Accept-Ranges: bytes
```

```
Content-Length: 35
```

```
Connection: close
```

```
Content-Type: text/html
```

```
<h1>My Home page</h1>
```

Status Line

Response Headers

Response
Message
Header

A blank line separates header & body

Response Message Body

Response

HTTP Response Codes

- ❑ Τριψήφιος Αριθμός όπου το πρώτο ψηφίο αντιστοιχεί σε μία κατηγορία
- ❑ **1xx** αντιστοιχεί σε ένα μήνυμα **πληροφόρησης**
- ❑ **2xx** αντιστοιχεί σε **επιτυχία**
 - 200 OK – Ο server πιστεύει ότι όλα είναι εντάξει
- ❑ **3xx** κάνει **redirect** τον client σε άλλο URL
 - 301 Moved Permanently
 - 302 Moved Temporarily
- ❑ **4xx** αντιστοιχεί σε κάποιο **λάθος** από την πλευρά του **client**
 - 404 Not Found – Το πιο διάσημο!
- ❑ **5xx** αντιστοιχεί σε **λάθος** από την πλευρά του **server**
 - 500 Server Error – Θα το βλέπετε συνέχεια

All HTTP codes

1XX Informational		4XX Client Error Continued	
100	Continue	409	Conflict
101	Switching Protocols	410	Gone
102	Processing	411	Length Required
2XX Success		412	Precondition Failed
200	OK	413	Payload Too Large
201	Created	414	Request-URI Too Long
202	Accepted	415	Unsupported Media Type
203	Non-authoritative Information	416	Requested Range Not Satisfiable
204	No Content	417	Expectation Failed
205	Reset Content	418	I'm a teapot
206	Partial Content	421	Misdirected Request
207	Multi-Status	422	Unprocessable Entity
208	Already Reported	423	Locked
226	IM Used	424	Failed Dependency
3XX Redirectional		426	Upgrade Required
300	Multiple Choices	428	Precondition Required
301	Moved Permanently	429	Too Many Requests
302	Found	431	Request Header Fields Too Large
303	See Other	444	Connection Closed Without Response
304	Not Modified	451	Unavailable For Legal Reasons
305	Use Proxy	499	Client Closed Request
307	Temporary Redirect	5XX Server Error	
308	Permanent Redirect	500	Internal Server Error
4XX Client Error		501	Not Implemented
400	Bad Request	502	Bad Gateway
401	Unauthorized	503	Service Unavailable
402	Payment Required	504	Gateway Timeout
403	Forbidden	505	HTTP Version Not Supported
404	Not Found	506	Variant Also Negotiates
405	Method Not Allowed	507	Insufficient Storage
406	Not Acceptable	508	Loop Detected
407	Proxy Authentication Required	510	Not Extended
408	Request Timeout	511	Network Authentication Required
		599	Network Connect Timeout Error
HTTP STATUS CODES			
When a browser requests a service from a web server, an error may occur.			
This is a list of HTTP status messages that might be returned.			



4xx HTTP codes

Σφάλματα πελάτη

❑ 400 Bad Request

- Το αίτημα δεν μπορεί να ολοκληρωθεί λόγω κακής σύνταξης του αιτήματος

❑ 401 Unauthorized

- Το αίτημα ήταν νόμιμο, αλλά ο server αρνείται να απαντήσει σε αυτό το αίτημα. Χρησιμοποιείται όταν ο χρήστης έχει τη δυνατότητα να συνδεθεί με κωδικό πρόσβασης και όνομα χρήστη, αλλά δεν τα έχει καταφέρει ακόμα ή όταν δεν έχει προσπαθήσει να συνδεθεί.

❑ 402 Payment Required

- Δεσμευμένο για μελλοντική χρήση

❑ 403 Forbidden

- Το αίτημα ήταν νόμιμο, αλλά ο server αρνείται να απαντήσει σε αυτό το αίτημα.

❑ 404 Not Found

- Η σελίδα που ζήτησε ο web browser δεν υπάρχει, αλλά μπορεί να είναι διαθέσιμη ξανά κάποια στιγμή στο μέλλον.

❑ 405 Method Not Allowed

- Το αίτημα που έγινε από μια σελίδα χρησιμοποιώντας μια μέθοδο αίτησης που δεν υποστηρίζεται από αυτή τη σελίδα.

❑ 406 Not Acceptable

- Ο server μπορεί να παράγει μια απάντηση που δεν είναι αποδεκτή από τον web browser

❑ 407 Proxy Authentication Required

- Ο πελάτης πρέπει πρώτα να συνδεθεί στον Proxy του

❑ 408 Request Timeout

- Ο server έληξε το αίτημα γιατί περίμενε πολύ ώρα για την απάντηση.

4xx HTTP codes

Σφάλματα πελάτη

❑ 409 Conflict

- Το αίτημα δεν μπορεί να ολοκληρωθεί λόγω μιας διένεξης στο αίτημα.

❑ 410 Gone

- Η σελίδα που ζήτησε ο web browser δεν είναι άλλο ποια διαθέσιμη

❑ 411 Length Request

- Δεν έχει οριστεί η παράμετρος "Content-Length". Ο Server δεν μπορεί να δεχτεί το αίτημα χωρίς αυτή την τιμή.

❑ 412 Precondition Failed

- Η προϋπόθεση που δόθηκε στο αίτημα θεωρήθηκε ψευδής από το server.

❑ 413 Request Entity Too Large

- Ο Server δεν μπορεί να δεχτεί το αίτημα, γιατί η οντότητα αίτησης είναι πολύ μεγάλη.

❑ 414 Request-URI Too Long

- Ο Server δεν θα αποδεχτεί το αίτημα, γιατί το URL είναι πολύ μακρύ. Αυτό συνήθως συμβαίνει όταν μετατρέπτε ένα αίτημα POST σε αίτημα GET με πολλές πληροφορίες ερωτήματος

❑ 415 Unsupported Media Type

- Ο Server δεν θα αποδεχτεί το αίτημα, γιατί ο τύπος των πολυμέσων δεν υποστηρίζεται.

❑ 416 Requested Range Not Satisfiable

- Ο πελάτης έχει ζητήσει ένα μέρος του αρχείου, αλλά ο server δεν μπορεί παρέχει αυτό το τμήμα.

5xx HTTP codes

Σφάλματα server

❑ 500 Internal Server Error

- Ένα σφάλμα γενικής σημασίας, το οποίο δημιουργείται όταν δεν υπάρχουν αρκετές πληροφορίες για το σφάλμα αυτό.

❑ 501 Not Implemented

- Ο Server είτε δεν αναγνωρίζει τη μέθοδο του αιτήματος ή δεν μπορεί να ολοκληρώσει το αίτημα.

❑ 502 Bad Gateway

- Ο Server λειτουργεί ως πύλη ή σαν Proxy και έλαβε μια άκυρη απάντηση από τον upstream server.

❑ 503 Service Unavailable

- Ο Server είναι προσωρινά μη διαθέσιμος (υπερφορτωμένος ή έχει πέσει).

❑ 504 Gateway Timeout

- Ο Server λειτουργεί ως πύλη ή σαν Proxy και δεν έχει λάβει απάντηση σε ένα εύλογο χρονικό διάστημα από τον upstream server.

❑ 505 HTTP Version Not Supported

- Ο Server δεν υποστηρίζει την έκδοση του HTTP πρωτοκόλλου που χρησιμοποιεί η αίτηση

❑ 511 Network Authentication Required

- Ο πελάτης πρέπει να συνδεθεί με κωδικό πρόσβασης και όνομα χρήστη για να αποκτήσει πρόσβαση

Servlets

Servlets

- ❑ Τα Servlets είναι ένα δημοφιλές πρωτόκολλο (ανεξάρτητο από την πλατφόρμα) και χρησιμοποιούνται σε server side
 - σε γλώσσα Java
- ❑ Χρησιμοποιείται σε java-enabled servers
 - e.g. tomcat, glassfish, jetty
- ❑ Έχει πρόσβαση σε όλα τα JAVA APIs
 - JDBC, other libraries
- ❑ Ουσιαστικά αποτελεί ένα πρόγραμμα Java, που τρέχει κάτω από έναν web server

Είναι σαν ένα διαμεσολαβητή μεταξύ ενός request από έναν web client και μία βάση δεδομένων/άλλη εφαρμογή στο server side

Servlets Tasks

- ❑ 1. Διάβασμα άμεσων δεδομένων από τους πελάτες
 - δεδομένα από αίτημα φόρμας / AJAX
- ❑ 2. Διάβασμα έμμεσων δεδομένα που αποστέλλονται από τους πελάτες
 - cookies, media types, compression schemes και τα λοιπά.
- ❑ 3. Επεξεργασία δεδομένων και δημιουργία αποτελεσμάτων
 - συνομιλία με βάση δεδομένων, υπηρεσίες Web Services, δημιουργία απάντησης
- ❑ 4. Αποστολή απάντησης
 - Το έγγραφο HTML, τα δεδομένα
- ❑ 5. Αποστολή μεταδεδομένων
 - Κεφαλίδες της απάντησης

Servlet containers

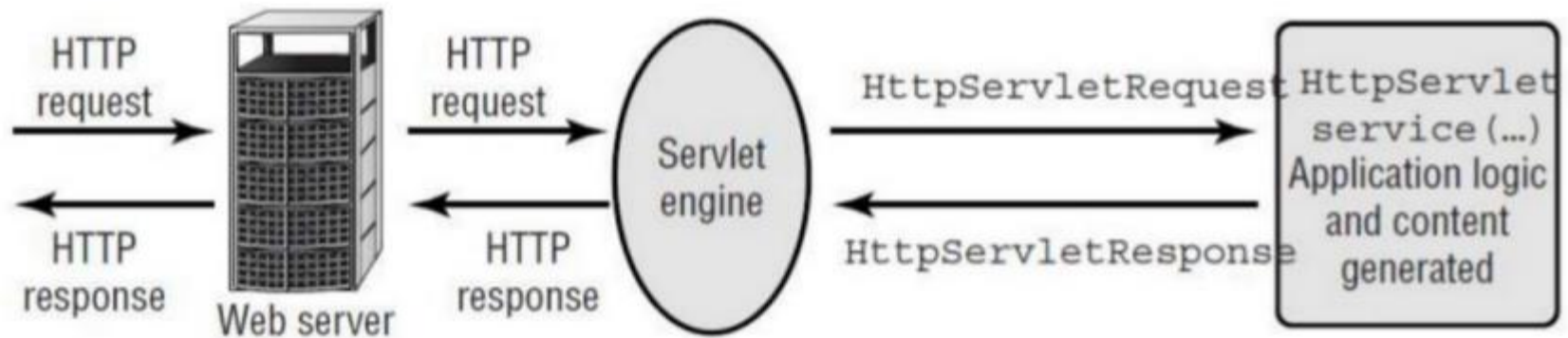
- ❑ Ένα servlet είναι java class και έτσι πρέπει εκτελείται σε Java virtual machine
- ❑ Ένα servlet container χρησιμοποιείται για αλληλεπίδραση/εκτέλεση Java servlets Java & JSP
- ❑ Ένα servlet container είναι υπεύθυνο για
 - διαχείριση του κύκλου ζωής των servlets
 - αντιστοίχιση διευθύνσεων URL σε ένα συγκεκριμένο servlet
 - δικαιώματα πρόσβασης για μια συγκεκριμένη διεύθυνση URL
 - Διαχείριση της δεξαμενής των threads



The request/response path

- ❑ Τυπικά ένα servlet
 - Λαμβάνει ένα request object `HttpServletRequest`
 - Επιστρέφει ένα response object `HttpServletResponse` μετά από κάποια προεργασία
- ❑ Ο container μετατρέπει
 - ένα HTTP request σε `HttpServletRequest` object
 - Και ένα `HttpServletResponse` object σε HTTP response

The request/response path



init(ServletConfig config)

- ❑ init method is executed using the ServletConfig object
- ❑ Causes the servlet implementing class GenericServlet to invoke its init version
 - init is called only once - not per each request
- ❑ Useful for initializing variables, opening DB connections, etc

service(HttpServletRequest req, HttpServletResponse res)

- ❑ service method is called in a new thread by server for each request
- ❑ Requires that the servlet is active, i.e. init()
- ❑ Dispatches the doGet and doPost methods based on the type of the request
- ❑ Usually no need to override this method!
 - support for HEAD, OPTIONS and TRACE requests
 - it does more things than just dispatching doGet and doPost

doGet and doPost

- ❑ `doGet(HttpServletRequest req, HttpServletResponse res)`
and `doPost(HttpServletRequest req, HttpServletResponse res)`
- ❑ methods that handle GET and POST requests respectively
 - objects for requests and responses are provided in the parameters
- ❑ Require an active servlet, i.e. `init()`
- ❑ Dispatched from `service()`
- ❑ Override these to provide desired behaviour
- ❑ Exceptions are send in the event of transmission or streaming errors

doGet and doPost parameters

- ❑ By using the passed in parameters you can access and create the necessary information to complete the requested task
- ❑ HttpServletRequest req provides a handle to the original request
 - get query parameters, header data, method type, date information
- ❑ HttpServletResponse res provides a handle to construct the response
 - set header data and body to display
- ❑ For the rest methods there are corresponding doXXX methods like doDelete, etc.

destroy(ServletConfig config)

- ❑ Called by the servlet container to indicate to a servlet that the servlet is being taken out of service
 - if the container is shutdown
 - if we reload the webapp
- ❑ It is not called after servicing each request!
- ❑ Useful for saving data to disk, closing DB connections, etc.

Reading the Query part

- ❑ **String getParameter(String param)**

- Χρήση της μεθόδου όταν η παράμετρος έχει μία και μόνο τιμή

- ❑ **String[] getParameterValues(String param)**

- Χρήση της μεθόδου όταν η παράμετρος έχει πολλαπλές τιμές

- ❑ Και οι δυο μέθοδοι γυρνούν null αν η παράμετρος δεν υπάρχει

- ❑ String param: **case-sensitive!**

Reading the Query part

- ❑ Parameter names: case sensitive
- ❑ Parameter values: είναι strings!
- ❑ Πρέπει να γράψετε κώδικα για να μετατρέψετε τα string σε τύπους
- ❑ Θυμηθείτε και τον αυτόματο τρόπο μέσω JSON
- ❑ Πάντα κάντε validate τις τιμές των παραμέτρων πριν την επεξεργασία των δεδομένων
 - Οι χρήστες μπορεί να βάλουν διαφορετικές τιμές από αυτές που περιμένουμε σε μία φόρμα
 - Βέβαια αυτά μπορούν να ελεγχθούν και μέσω JavaScript και HTML 5 forms

Διάβασμα άλλων δεδομένων

Για Κείμενο, πχ JSON

- `request.getReader()` // for text

Για άλλα αρχεία

- `request.getInputStream()` // raw data

Δε στέλνονται απευθείας από φόρμες

- Τα επεξεργάζεστε μέσω AJAX όπως έχουμε δει και τα στέλνετε έπειτα (για JSON)

Τα δεδομένα δεν είναι προσβάσιμα μέσω της `request.getParameter`

Σύνοψη - Οδηγίες

- ❑ **Διαβάζουμε όσα είδαμε και κάνουμε σχετικά παραδείγματα**
- ❑ **Δεν είναι στην ύλη θέματα**
 - CSS, Android, Security Issues
 - Δείτε και τις σχετικές οδηγίες στο elearn
- ❑ **Επίσης δε χρειάζεται να εμβαθύνετε σε θέματα JSP και GRAPHQL**
 - Πρέπει όμως να ξέρετε τη βασική ιδέα
- ❑ **Κάθε φοιτητής θα μπορεί να έχει σημειώσεις μέχρι 5 κόλλες A4**
 - Μπρος-πίσω
- ❑ **Μπορείτε να τις γράψετε**
 - είτε σε κάποιο δικό σας doc και να τις εκτυπώσετε
 - είτε χειρόγραφες
- ❑ **Είναι σημαντικό να μην αγχώνεστε για την εξέταση**
 - Ακόμα και να μην πάει καλά, υπάρχει και η επαναληπτική εξέταση του Σεπτεμβρίου