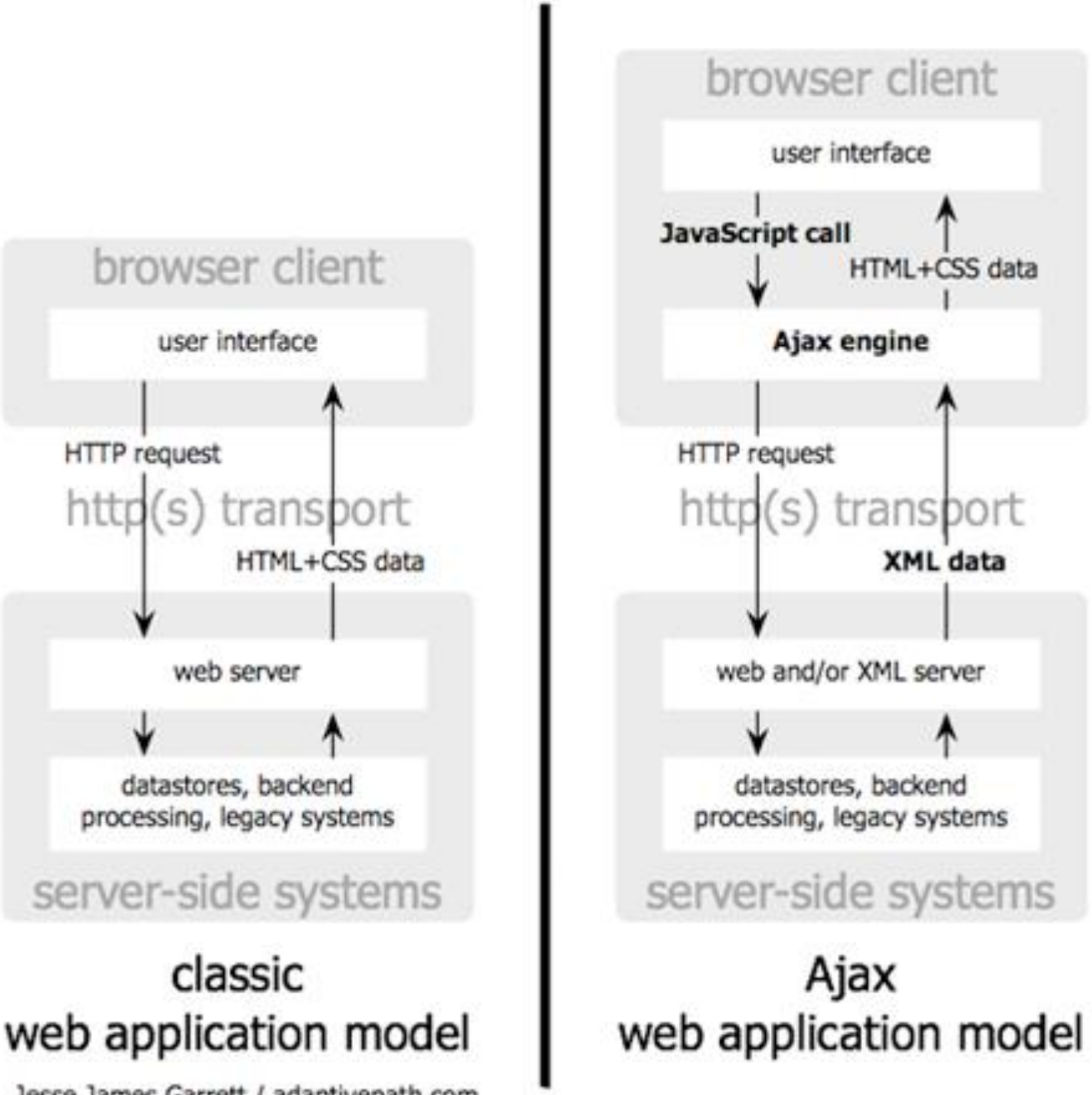
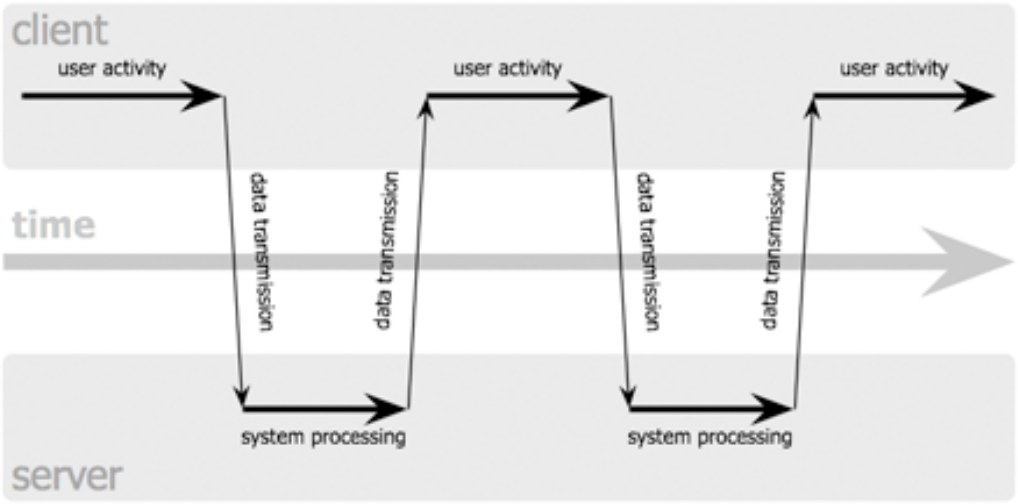
AJAX

AJAX

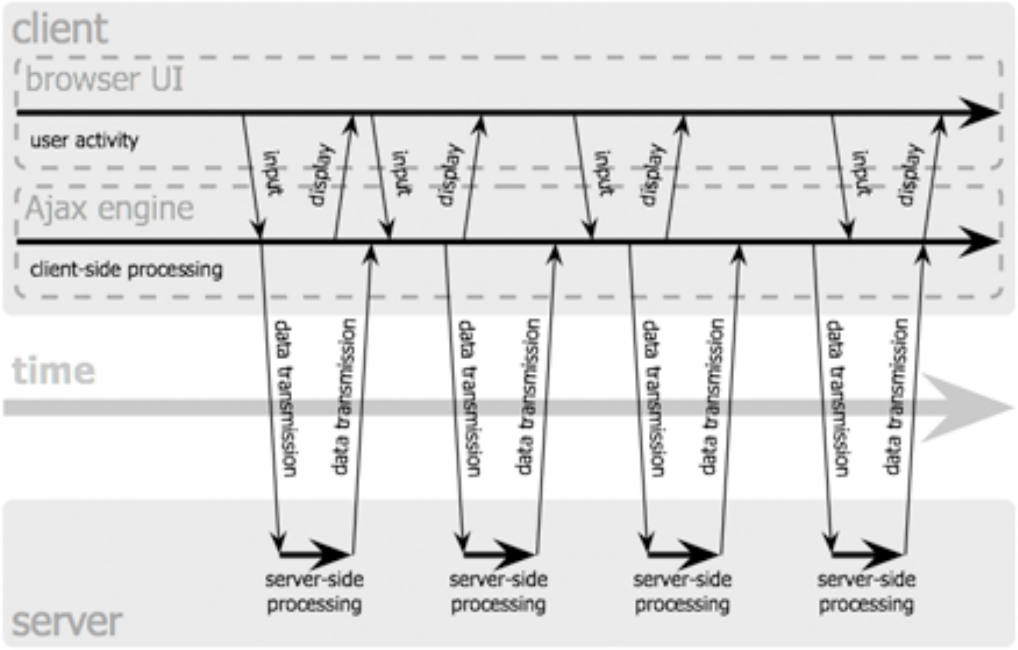
- AJAX είναι ένα νέο "trend" στο δικτυακό προγραμματισμό.
  - Είναι συνδυασμός client και server side προγραμματισμού.
- **AJAX = Asynchronous JavaScript and XML**
- Δεν πρόκειται για νέα γλώσσα αλλά τεχνική για γρήγορες αποδοτικές και πλήρως διαδραστικές εφαρμογές.
- Μέσω του **XMLHttpRequest** object ο JavaScript κώδικας που τρέχει στο browser μπορεί να επικοινωνεί με το server.
- Υπάρχουν αρκετές έτοιμες βιβλιοθήκες διαθέσιμες ανάλογα με το σκοπό:
  - πχ Google maps
  - οι οποίες είναι κυρίως κώδικας JavaScript.


browser client
user interface
HTTP request
http(s) transport
HTML+CSS data
web server
datastores, backend
processing, legacy systems
server-side systems
classic
web application model
Jesse James Garrett / adaptivepath.com
browser client
user interface
JavaScript call
HTML+CSS data
Ajax engine
HTTP request
http(s) transport
XML data
web and/or XML server
datastores, backend
processing, legacy systems
server-side systems
Ajax
web application model

classic web application model (synchronous)



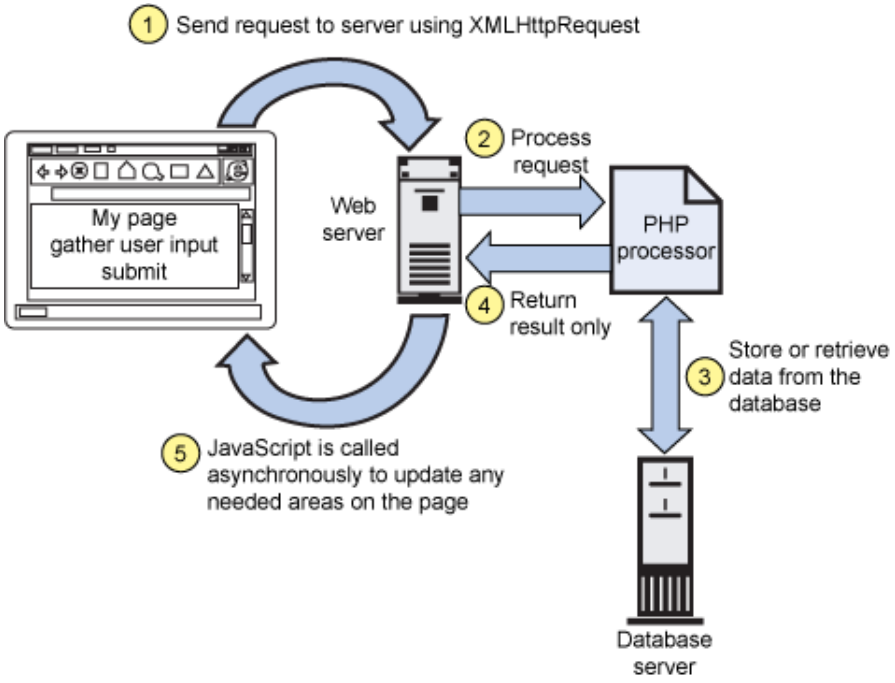
Ajax web application model (asynchronous)



Jesse James Garrett / adaptivepath.com

Χρησιμότητα

- Αυξάνει τη λειτουργικότητα και τις δυνατότητες των web εφαρμογών
- Μπορούμε να έχουμε Rich Internet Applications χωρίς Flash
- Εξοικονόμηση Bandwidth
  - Κατεβάζουμε μόνο τα δεδομένα που θέλουμε.
- Ασύγχρονη επικοινωνία με το Server



Μειονεκτήματα

- Χαλάει το “back button support”.
  - Υπάρχει όμως τρόπος να το διορθώσουμε
- Τα URLs δεν αλλάζουν καθώς αλλάζει η σελίδα.
  - Υπάρχει όμως τρόπος να το διορθώσουμε
- Μπορεί να υπάρχει ασυμβατότητα μεταξύ των browsers.
  - Δεν υπάρχει πρόβλημα αν χρησιμοποιούμε κάποιο framework/library (πχ. Jquery)
- Η Javascript μπορεί να καταναλώνει πολύ υπολογιστική ισχύ (ειδικά σε μικρές συσκευές κινητά-tablets).
- Το Debugging είναι αρκετά δύσκολο.

XMLHttpRequest

- Παραδοσιακά, για να ανακτήσουμε ή να στείλουμε πληροφορία από ή προς τον server έπρεπε να έχουμε φόρμες, και να παρουσιάσουμε τα αποτελέσματα σε μια νέα σελίδα.
- Η διαδικασία απλοποιείται με το XMLHttpRequest μέσω το οποίου η Javascript μπορεί να επικοινωνήσει με το server χωρίς να απαιτείται να φορτωθεί πάλι η σελίδα.
- Είναι ένα object που υποστηρίζεται από όλους τους σύγχρονους browsers (Internet Explorer, Firefox, Chrome, Opera, and Safari).
  - Δυστυχώς όχι πάντα με τον ίδιο τρόπο…

Κύριες Ιδιότητες του XMLHttpRequest

- readyState
  - 0 = UNINITIALIZED
    - όταν δεν το έχουμε καλέσει ακόμη
  - 1 = LOADING
    - έχει αρχικοποιηθεί το request
  - 2 = LOADED
    - έχει γίνει το request
  - 3 = INTERACTIVE
    - πραγματοποιείται λήψη των δεδομένων
  - 4 = COMPLETED
    - ολοκληρώθηκε
- responseText
  - το response σαν κείμενο ή null αν το υπάρχει λάθος ή το readystate < 3
- responseXML
  - το response σαν DOM Document object ή null αν το υπάρχει λάθος ή το readystate < 3
- onreadystatechange
  - Διατηρεί το όνομα της συνάρτησης που θα καλείται κάθε φορά που αλλάζει το readyState

Κύριες Μέθοδοι του XMLHttpRequest

- open(method, url, async)
  - Αρχικοποιεί ένα XMLHttpRequest
  - Το method μπορεί να είναι GET, POST
  - Το url που θα γίνει το request
  - Το async είναι boolean που δείχνει αν το request θα σταλεί asynchronously
    - Το default είναι true
- send(body)
  - Στέλνει το request
  - Το body είναι null εάν χρησιμοποιούμε GET
- abort()
  - Ακυρώνει το request

Δημιουργία XMLHttpRequest

- Για σύγχρονους browsers:
  - *xmlhttp=new XMLHttpRequest()*
- Για IE5 ή IE6
  - *xmlhttp=new ActiveXObject("Microsoft.XMLHTTP")*
  - *Πλέον είναι απίθανο κάποιος χρήστης να έχει IE6.*

- Επομένως για να καλύψουμε και τις δύο περιπτώσεις:

```
…
if (window.XMLHttpRequest) {
    // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
} else {
    // code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
…
```

Για συγχρονισμένη κλήση

- Έστω ότι θέλουμε να αλλάξουμε την τιμή του html element με id=“test”με τα περιεχόμενα ενός αρχείου που υπάρχει στο server μόλις πατηθεί ένα button.
- Θα δημιουργήσουμε μια συνάρτηση που θα «φορτώνει» το αρχείο. Έστω: *loadDoc()*

- Το html element (div):
  *<div id="test“><h2>Click to let AJAX change this text</h2> *
- Το button:
  <input type="submit" value="Click" onclick="loadDoc('test1.txt')"/>
- Στο head:

```
<script type="text/javascript">
    function loadDoc(url)
    {
      //μόνο για IE7+, Mozilla, Opera klt…
      xmlhttp=new XMLHttpRequest();
      xmlhttp.open("GET",url,false);
      xmlhttp.send(null);
      document.getElementById('test').innerHTML=xmlhttp.responseText;
    }
</script>
```

Για συγχρονισμένη κλήση - Παράδειγμα

```
<html>
<head>
<script type="text/javascript">
function loadDoc(url) {
        //μόνο για IE7+, Mozilla, Opera klt…
        xmlhttp=new XMLHttpRequest();
        xmlhttp.open("GET",url,false);
        xmlhttp.send(null);
        document.getElementById('test').innerHTML=xmlhttp.responseText;
}
</script>
</head>
        <body>
                <div id="test">
                                <h2>Click to let AJAX change this text</h2>
                </div>
                <input type="submit" value="Click" onclick="loadDoc('test1.txt')"/>
        </body>
</html>
```

Για Ασύγχρονη κλήση (1)

- Για ασύγχρονη κλήση θα πρέπει να ελέγχουμε την κατάσταση του request μέσα από τη μεταβλητή readyState.
- Η πιο απλή λύση είναι να δημιουργήσουμε μια νέα συνάρτηση (έστω onComplete) που θα καλείται όταν αλλάζει κατάσταση η readyState.

```
function onComplete() {
    if (xmlhttp.readyState==4) {
      // 4 = "loaded“
    document.getElementById('test').innerHTML=xmlhttp.responsetext;
    }
}
```

Για Ασύγχρονη κλήση - Παράδειγμα

```
<html>
<head>
<script type="text/javascript">
function loadDoc(url)
{
xmlhttp=new XMLHttpRequest();
xmlhttp.onreadystatechange=onComplete;
xmlhttp.open("GET",url,true);
xmlhttp.send(null);
}

function onComplete() {
  if (xmlhttp.readyState==4) {
       document.getElementById('test').innerHTML=
                  xmlhttp.responseText;
  }
}
</script>
....
....
</head>
<body>
<div id="test">
<h2>Click to let AJAX change this text</h2>
</div>
<input type="submit" value="Click"
       onclick="loadDoc('test1.txt')"/>
</body>
</html>
```

1.2 Εφαρμογές

- Η χρήση του AJAX προσδίδει την αίσθηση μιας εφαρμογής που τρέχει στον υπολογιστή του χρήστη στις εφαρμογές ιστού.
- Παραδείγματα αποτελούν το google maps και το gmail

1.3 XMLHttpRequest

- Το συγκεκριμένο αντικείμενο αποτελεί τον πυρήνα της τεχνικής.
- Μπορεί να στέλνει *ασύγχρονα* αιτήσεις στον εξυπηρετητή και να ορίζει κάποια συνάρτηση που θα καλείται, όταν έρθει η απάντηση.
- Μέσω του χειρισμού του DOM της HTML ανανεώνεται η ιστοσελίδα που βλέπει ο χρήστης δυναμικά με τα αποτελέσματα της απάντησης

1.4 Επικοινωνία με τον Εξυπηρετητή

- Η αίτηση που θα αποσταλεί μπορεί να είναι τύπου GET ή POST (συνήθως)
- Για λόγους ασφάλειας, επικοινωνία μπορεί να γίνει μόνο με τον εξυπηρετητή που φιλοξενεί την ιστοσελίδα.
- Επίσης, υποστηρίζεται και σύγχρονη επικοινωνία.

1.5 XML

- Μία από τις λέξεις που αποτελούν το ακρωνύμιο AJAX είναι η XML, που προβλέπεται να χρησιμοποιηθεί για την ανταλλαγή δεδομένων μεταξύ πελάτη και εξυπηρετητή
- Ωστόσο, η χρήση της όχι μόνο δεν είναι απαραίτητη, αλλά ορισμένες φορές ούτε καν συνιστώμενη
- Μπορεί να χρησιμοποιηθεί μία απλή συμβολοσειρά

1.6 Συμβατότητα

- Πρόκειται για αρκετά νέα τεχνική, ως εκ τούτου δεν υποστηρίζεται πολύ καλά (έως καθόλου) από παλαιότερους περιηγητές (ή υποστηρίζεται με διαφορετικό τρόπο)
- Τα παραδείγματα που ακολουθούν υποστηρίζονται (μεταξύ άλλων) από τους περιηγητές Internet Explorer 7 και Mozilla Firefox 2 και μεταγενέστερες εκδόσεις αυτών

1.8 Παράδειγμα

- Η τεχνική AJAX απαιτεί τη συνεργασία:
  - HTML
  - JavaScript
  - Εξυπηρετητή

HTML

- Έστω το επόμενο έγγραφο HTML:

```
<html>
  <head>
    <title>AJAX</title>
  </head>
  <body>
    <form>
      <input type=“text” id=“username”
           onchange=“getUsername();”>
      <p id=“name”></p>
    </form>
  </body>
</html>
```

Εξυπηρετητής

- Ο εξυπηρετητής έχει το αρχείο getname.php με περιεχόμενα

```
<?php
$username = $_GET[“username”];
if ($username == “user1”) {
    print “USER1”;
} else {
    print “UNKNOWN USER”;
}
?>
```

JavaScript – Αποστολή Δεδομένων

- Την αποστολή των δεδομένων αναλαμβάνει το επόμενο τμήμα κώδικα:

```
xmlHttp = new XMLHttpRequest();
function getUsername() {
    var username =
        document.getElementById("username").value;
    if (username == null || username == "") {
        return;
    }
    var url = "/getname.php?username=" +  username;
    xmlHttp.open("GET", url, true);
    xmlHttp.onreadystatechange = setName;
    xmlHttp.send(null);
}
```

Κώδικας

```
xmlHttp = new XMLHttpRequest();
function getUsername() {
    var username =
      document.getElementById("username").value;
    if (username == null || username == "") {
        return;
    }
    var url = "/getname.php?username=" +  username;
    xmlHttp.open("GET", url, true);
    xmlHttp.onreadystatechange = setName;
    xmlHttp.send(null);
}
```

Επεξήγηση

- Αρχικά δημιουργείται το αντικείμενο `xmlHttp` τύπου `XMLHttpRequest`

Κώδικας

```
xmlHttp = new XMLHttpRequest();
function getUsername() {
    var username =
        document.getElementById("username").value;
    if (username == null || username == "") {
        return;
    }
    var url = "/getname.php?username=" +  username;
    xmlHttp.open("GET", url, true);
    xmlHttp.onreadystatechange = setName;
    xmlHttp.send(null);
}
```

Επεξήγηση

- Ακολούθως ορίζεται η συνάρτηση που θα κληθεί, όταν ο χρήστης αλλάξει την τιμή του πεδίου εισόδου κειμένου στον περιηγητή, η συνάρτηση `getUsername()`.

Κώδικας

```
xmlHttp = new XMLHttpRequest();
function getUsername() {
    var username =
        document.getElementById("username").value;
    if (username == null || username == "") {
        return;
    }
    var url = "/getname.php?username=" +  username;
    xmlHttp.open("GET", url, true);
    xmlHttp.onreadystatechange = setName;
    xmlHttp.send(null);
}
```

Επεξήγηση

- Αρχικά, διαβάζεται το κείμενο που πληκτρολόγησε ο χρήστης στο συγκεκριμένο πεδίο εισόδου κειμένου.
- Για το σκοπό αυτό χρησιμοποιείται η συνάρτηση `getElementById` του αντικειμένου document (που αναπαριστά το έγγραφο).
- Από το επιστρεφόμενο αντικείμενο αποθηκεύεται η συμβολοσειρά που έχει εισαγάγει ο χρήστης (ιδιότητα `value`) στη μεταβλητή `username`.

Κώδικας

- Την αποστολή των δεδομένων αναλαμβάνει το επόμενο τμήμα κώδικα:

```
xmlHttp = new XMLHttpRequest();
function getUsername() {
    var username =
        document.getElementById("username").value;
    if (username == null || username == "") {
        return;
    }
    var url = "/getname.php?username=" +  username;
    xmlHttp.open("GET", url, true);
    xmlHttp.onreadystatechange = setName;
    xmlHttp.send(null);
}
```

Επεξήγηση

- Στη συνέχεια γίνεται έλεγχος, εάν ο χρήστης έχει γράψει κάτι ή εάν, αντίθετα, έχει αφήσει το πεδίο κενό, οπότε δεν απαιτείται να γίνει κάποια ενέργεια

Κώδικας

```
xmlHttp = new XMLHttpRequest();
function getUsername() {
    var username =
        document.getElementById("username").value;
    if (username == null || username == "") {
        return;
    }
    var url = "/getname.php?username=" +  username;
    xmlHttp.open("GET", url, true);
    xmlHttp.onreadystatechange = setName;
    xmlHttp.send(null);
}
```

Επεξήγηση

- Κατόπιν σχηματίζεται το κατάλληλο url.
- Το url αποτελείται από το όνομα του αρχείου και την παράμετρο username με την τιμή της μεταβλητής `username` (μορφή παραμέτρου σε αίτηση τύπου GET).

Κώδικας

```
xmlHttp = new XMLHttpRequest();
function getUsername() {
    var username =
        document.getElementById("username").value;
    if (username == null || username == "") {
        return;
    }
    var url = "/getname.php?username=" +  username;
    xmlHttp.open("GET", url, true);
    xmlHttp.onreadystatechange = setName;
    xmlHttp.send(null);
}
```

Επεξήγηση

- Με τη μέθοδο **open()** ανοίγεται μία σύνδεση με τον εξυπηρετητή.
- Οι παράμετροί της δηλώνουν ότι:
  - Η αίτηση είναι τύπου GET (διαφορετικά, θα μπορούσε να είναι POST)
  - Το url της αίτησης
  - Η επικοινωνία θα γίνει ασύγχρονα

Κώδικας

```
xmlHttp = new XMLHttpRequest();
function getUsername() {
    var username =
        document.getElementById("username").value;
    if (username == null || username == "") {
        return;
    }
    var url = "/getname.php?username=" +  username;
    xmlHttp.open("GET", url, true);
    xmlHttp.onreadystatechange = setName;
    xmlHttp.send(null);
}
```

Επεξήγηση

- Η συγκεκριμένη δήλωση ορίζει τη συνάρτηση που θα κληθεί, όταν έρθει η απάντηση του εξυπηρετητή. Η συνάρτηση αυτή θα εκτελεστεί, δηλαδή, όταν έρθει η απάντηση από τον εξυπηρετητή, και είναι υπεύθυνη για το χειρισμό και επεξεργασία αυτών. Για παράδειγμα, η ανανέωση της ιστοσελίδας με τα νέα δεδομένα.

Κώδικας

```
xmlHttp = new XMLHttpRequest();
function getUsername() {
    var username =
        document.getElementById("username").value;
    if (username == null || username == "") {
        return;
    }
    var url = "/getname.php?username=" +  username;
    xmlHttp.open("GET", url, true);
    xmlHttp.onreadystatechange = setName;
    xmlHttp.send(null);
}
```

Επεξήγηση

- Η συγκεκριμένη δήλωση αποστέλλει την αίτηση με τα δεδομένα της στον εξυπηρετητή

JavaScript – Λήψη Δεδομένων

- Κατά τη δημιουργία της αίτησης προς τον εξυπηρετητή ορίστηκε ότι η συνάρτηση που πρέπει να κληθεί, μόλις ληφθεί η απάντηση, είναι η συνάρτηση `setName()`.

Κώδικας

```
function setName() {
    if (xmlHttp.readyState == 4) {
        if (xmlHttp.status == 200) {
            var response =
                xmlHttp.responseText;
            var name =
                document.getElementById("name");
            name.innerHTML = "Το όνομα είναι " +
                response;
        } else {
            alert("Σφάλμα");
        }
    }
}
```

Κώδικας

```
function setName() {
    if (xmlHttp.readyState == 4) {
        if (xmlHttp.status == 200) {
            var response =
                xmlHttp.responseText;
            var name =
                document.getElementById("name");
            name.innerHTML = "Το όνομα είναι " +
                response;
        } else {
            alert("Σφάλμα");
        }
    }
}
```

Επεξήγηση

- Στην αρχή ορίζεται ότι ο υπόλοιπος κώδικας θα εκτελεστεί, μόνο όταν η κατάσταση

Κώδικας

```
function setName() {
    if (xmlHttp.readyState == 4) {
        if (xmlHttp.status == 200) {
            var response =
                xmlHttp.responseText;
            var name =
                document.getElementById("name");
            name.innerHTML = "Το όνομα είναι " +
                response;
        } else {
            alert("Σφάλμα");
        }
    }
}
```

Επεξήγηση

- Το πρωτόκολλο HTTP ορίζει διάφορους κωδικούς αριθμούς που αναφέρονται σε μία απάντηση για κάποια αίτηση που υποδηλώνουν είτε ότι ο εξυπηρετητής εκτέλεσε επιτυχώς την αίτηση (αριθμός 200) είτε ότι συνέβη κάποιο σφάλμα κατά την επεξεργασία της αίτησης από τον εξυπηρετητή.
- Εδώ δηλώνεται ότι, αν δεν υπήρξε σφάλμα, να εκτελεστεί το επόμενο τμήμα κώδικα. Διαφορετικά, να ειδοποιηθεί ο χρήστης για το σφάλμα.

Κώδικας

```
function setName() {
    if (xmlHttp.readyState == 4) {
        if (xmlHttp.status == 200) {
            var response =
                xmlHttp.responseText;
            var name =
                document.getElementById(“name”);
            name.innerHTML = “Το όνομα είναι “ +
                response;
        } else {
            alert(“Σφάλμα”);
        }
    }
}
```

Επεξήγηση

- Η μεταβλητή `xmlHttp` αντιστοιχεί στην ίδια μεταβλητή με πριν (καθολική μεταβλητή).
- Σε αυτήν περιέχεται και η απάντηση, η οποία βρίσκεται στην ιδιότητα `responseText` με τη μορφή συμβολοσειράς.

Κώδικας

```
function setName() {
    if (xmlHttp.readyState == 4) {
        if (xmlHttp.status == 200) {
            var response =
                xmlHttp.responseText;
            var name =
                document.getElementById("name");
            name.innerHTML = "Το όνομα είναι " +
                response;
        } else {
            alert("Σφάλμα");
        }
    }
}
```

Επεξήγηση

- Στη συνέχεια, ορίζεται η μεταβλητή `name` που αντιστοιχεί στο στοιχείο της HTML με τιμή για την παράμετρο id τη “name”.
- Αποθηκεύεται στην ιδιότητα `innerHTML` της μεταβλητής αυτής η απάντηση από τον εξυπηρετητή.

Αποτέλεσμα

- Κάθε φορά που ο χρήστης πληκτρολογεί κάποιο όνομα χρήστη στο πεδίο εισόδου κειμένου της ιστοσελίδας (μετά από την επικοινωνία με τον εξυπηρετητή) εμφανίζεται στην ιστοσελίδα του όνομα του αντίστοιχου χρήστη.