# Javascript

AJAX

(Asychronous Javascript and XML)

Νικόλαος Κατσάλης

29/5/2026

# Περιεχόμενα

- Τι είναι το AJAX
- Πλεονεκτήματα και χρήσεις
- Τρόποι Υλοποίησης (XMLHttpRequest, fetch API)

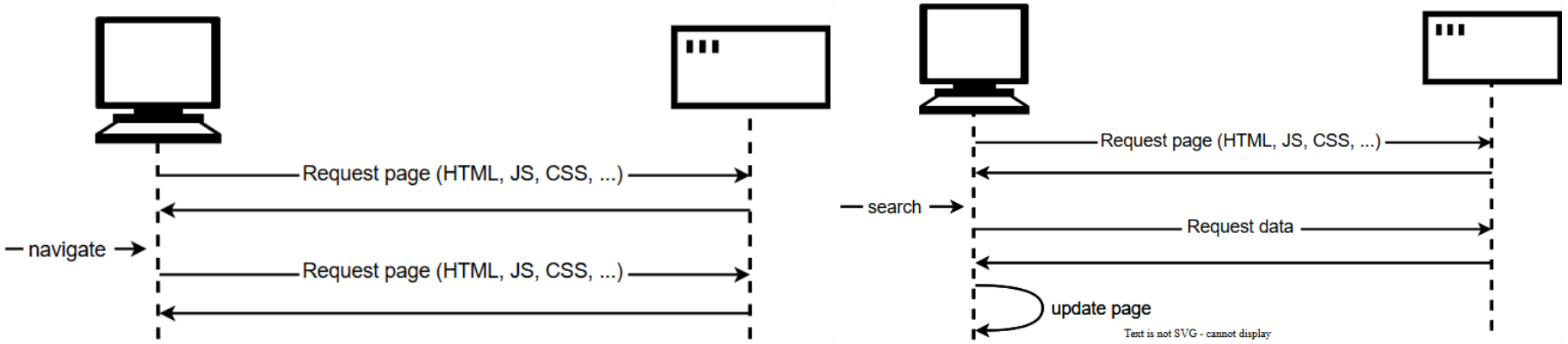
# Τι είναι το AJAX

- Asynchronous JavaScript and XML (**Ajax** ή **AJAX**) είναι μία τεχνική ανάπτυξης web εφαρμογών με την οποία μία σελίδα που προβάλλεται σε έναν φυλλομετρητή, εκτελώντας Javascript κώδικα κάνει **ασύγχρονα** HTTP requests, παίρνει τα αποτελέσματα και τα προβάλλει σε διάφορα μέρη της ιστοσελίδας χωρίς να απαιτείται η πλήρης επαναφόρτωση της σελίδας από τον web server.

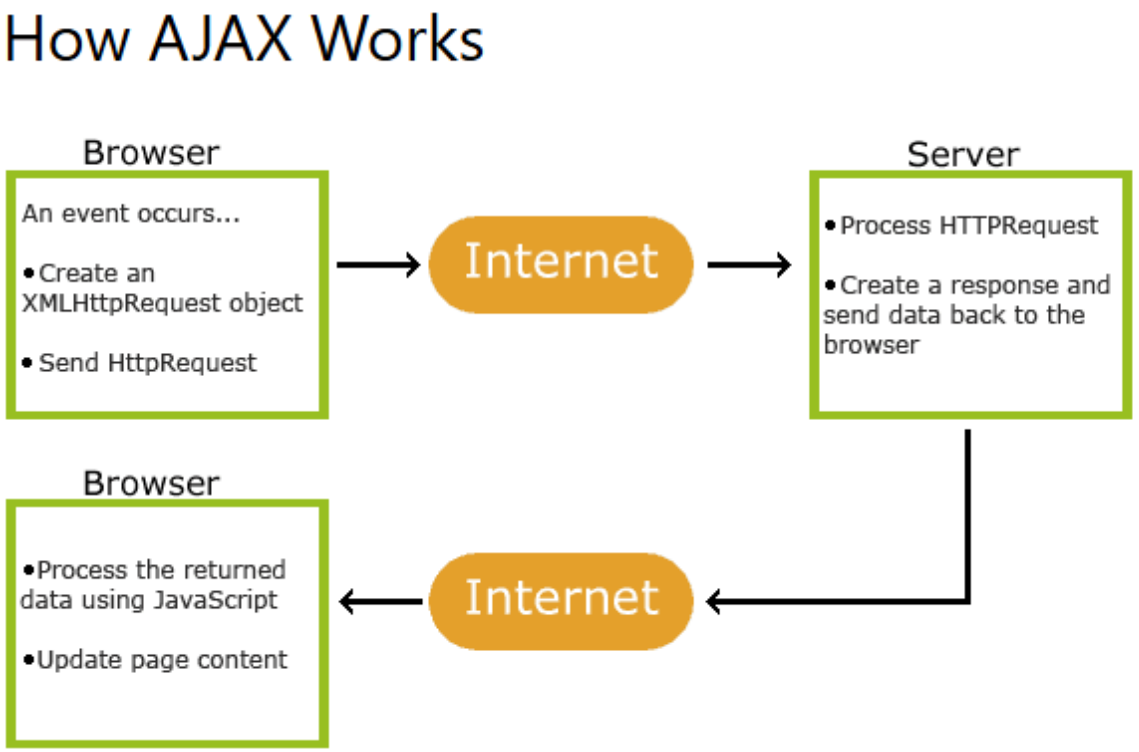
# Τι είναι το AJAX

**Κλασικός τρόπος επικοινωνίας**

**Ajax**



# Τι είναι το AJAX



# Τι είναι το AJAX

- Αυτή η τεχνική κάνει την ιστοσελίδα πιο responsive αφού στην σελίδα μας ενημερώνονται μόνο τα μέρη που μας ενδιαφέρουν.
- Οπότε με αυτή την τεχνική μπορούμε να φτιάξουμε εφαρμογές που αποτελούνται μόνο από μία σελίδα που χρησιμοποιεί Ajax για να ενημερώσει το περιεχόμενό της όπως απαιτείται κάθε φορά.

# Πλεονεκτήματα και χρήσεις

- Οι πιο συνηθισμένες χρήσεις είναι:
  1. Μεταφορά δεδομένων από τον web server αφού φορτωθεί η σελίδα.
  2. Ενημέρωση της σελίδας χωρίς την επαναφόρτωσή της.
  3. Αποστολή δεδομένων στον web server στο παρασκήνιο.

# Πλεονεκτήματα και χρήσεις

- Η τεχνική AJAX είναι ιδιαίτερα συνηθισμένη σε ιστοτόπους που εξαρτώνται πολύ από τα δεδομένα όπως π.χ. is a common Amazon, YouTube, eBay κλπ. Τα βασικά πλεονεκτήματα είναι:
  1. Οι σελίδες ενημερώνονται πολύ γρηγορότερα και δεν χρειάζεται να περιμένει ο χρήστης για την ανανέωση της ιστοσελίδας. Άρα η σελίδα γίνεται αντιληπτή ως γρηγορότερη και πιο responsive από τους χρήστες.
  2. Λιγότερα δεδομένα κατεβαίνουν σε κάθε ενημέρωση και άρα έχουμε χρήση μικρότερου bandwidth. Αυτό βοηθάει ιδιαίτερα σε συσκευές όπως smartphones κλπ αλλά και όπου το διαδίκτυο δεν είναι πολύ γρήγορο.

# Τρόποι Υλοποίησης

- Η τεχνική αυτή απαιτεί τη χρήση Javascript και υλοποιείται με δύο τρόπους:
  1. Με τη χρήση του αντικειμένου **XMLHttpRequest.** Είναι ο παλιός τρόπος υλοποίησης και έχει εφαρμοστεί στα περισσότερα υπάρχοντα projects.
  2. Με τη χρήση του **fetch API.** Ο τρόπος αυτός είναι πιο σύγχρονος, πιο γρήγορος και προτιμάται στους σύγχρονους φυλλομετρητές και προτείνεται για κάθε νέο project.

# Τρόποι Υλοποίησης - XMLHttpRequest

- Με τη χρήση του αντικειμένου XMLHttpRequest ακολουθούμε τα επόμενα βήματα:
  1. Δημιουργούμε στην Javacript ένα XMLHttpRequest object για να κάνει το request (για να στείλει ή/και πάρει δεδομένα)
  2. Ορίζουμε μία συνάρτηση που θα εκτελεστεί για να διαχειριστεί τα δεδομένα-απάντηση όταν ληφθούν
  3. Ανοίγουμε τη σύνδεση με το XMLHttpRequest object
  4. Στέλνουμε το Request στον server με το XMLHttpRequest object XMLHttpRequest object

# Τρόποι Υλοποίησης - XMLHttpRequest

```
// Δημιουργούμε το XMLHttpRequest object
var req= new XMLHttpRequest();

// Ορίζουμε τη συνάρτηση που θα διαχειριστεί την απάντηση
function  handleResponse() {
 ....
}
req.onload =  handleResponse;

// Ανοίγουμε τη σύνδεση και στέλνουμε το  request
req.open("GET", “/folder/file.php");
req.send();
```

# Τρόποι Υλοποίησης – Fetch API

- Με τη χρήση του Fetch API:
- Κάνουμε ένα request καλώντας τη fetch() η οποία είναι διαθέσιμη ως καθολική συνάρτηση στο αντικείμενο window. Όταν την καλούμε της περνάμε ένα Request object ή ένα string που περιέχει το url που θα φέρουμε μαζί με προαιρετικό όρισμα για να διαμορφώσουμε το request.
- Η συνάρτηση fetch() επιστρέφει ένα αντικείμενο Promise που περιέχει ένα Response object το οποίο έχει την απάντηση. Μπορούμε να ελέγξουμε την κατάσταση του request και να πάρουμε την απάντηση σε διάφορες μορφές (text, JSON κλπ) καλώντας την κατάλληλη μέθοδο στην απάντηση.

# Τρόποι Υλοποίησης - Fetch API

```
// Δημιουργούμε το XMLHttpRequest object
async function getData() {
        const url = "https://example.org/products.json";
        try {
          const response = await fetch(url);
          if (!response.ok) {
            throw new Error(`Response status: ${response.status}`);
          }
          const result = await response.json();
          console.log(result);
        }
        catch (error) {
          console.error(error.message);
        }
}

// Ορίζουμε τη συνάρτηση που θα διαχειριστεί την απάντηση
function  handleResponse() {
 ….
}
req.onload =  handleResponse;

// Ανοίγουμε τη σύνδεση και στέλνουμε το  request
req.open("GET", “/folder/file.php");
req.send();
```

# Αναφορές

- MDN Web Docs Mozilla (https://developer.mozilla.org/en-US/docs/Glossary/AJAX, https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch)
- W3Schools (https://www.w3schools.com/js/js_ajax_intro.asp)