Related
Dynamic Textarea height
Upload Progress Bar
Flask Lazy Loading

Recommended books
Python Basics
Advanced Python
Flask Web Development
Django Web Development
Python Machine Learning
Python Testing

Recommended learning
Pluralsight
Treehouse

Uploading files with a progress bar and percentage - XMLHTTPRequest

Providing visual feedback on the progress of uploading large files with a progress bar and percentage complete

Article Posted on 25 Feb 2019 by Julian Nash in JavaScript

flask javascript

User feedback is important, very important. If an application doesn't provide the user with the right level of feedback, there's high chance that they'll end up disliking it.

Posting a large file to the server using a form poses an immediate problem as it could take several minutes or even hours, depending on the filesize.

Instead, we should provide the user with some visual feedback, indicating the status of their upload.

In this example, we're going to create a single input element, allowing the user to select a file and upload it to the server. We're not concerned about what happens with the file once it gets to the server (We've covered that in another guide).

We're going to post the file to the server using JavaScript's XMLHttpRequest and keep them updated on the status with a progress bar and a percentage of how much of the file has been uploaded.

What we'll be building:

At this point I should probably point out that I'm not a JavaScript expert. I'm sure this code can be improved and optimized so feedback is welcome.

The route

We're using Python & Flask for our web server, but feel free to adapt this example to your own needs.

As discussed, we're not concerned about what happens to the file once it gets to the server, so we'll keep the route simple:

```
from flask import render_template, request, make_response, jsonify

@app.route("/upload-video", methods=["GET", "POST"])
def upload_video():

    if request.method == "POST":

        file = request.files["file"]

        print("File uploaded")
        print(file)

        res = make_response(jsonify({"message": "File uploaded"}), 200)

        return res

    return render_template("public/upload_video.html")
```

We're accessing the file with `request.files["file"]`, printing some information about the file and returning a simple JSON response with a `200` HTTP status code.

We're rendering a file called `upload_video.html`, so let's create it.

The HTML

We're using Bootstrap 4 in this example, but feel free to use your own styling.

```
<!doctype html>
<html lang="en">

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.cs

    <title>Upload video</title>
</head>

<body>

    <div class="container">
        <div class="row">
            <div class="col">

                <div class="mb-3 mt-3">

                    <h2 class="mb-3" style="font-weight: 300">Upload video</h2>

                    <div class="form-group mb-3">
                        <div class="custom-file">
                            <input type="file" class="custom-file-input" name="file_input" id="file_input" onInput=
                            <label id="file_input_label" class="custom-file-label" for="image">Select file</label>
                        </div>
                    </div>

                    <button onclick="upload({{ request.url }})" id="upload_btn" class="btn btn-primary">Uploa

                    <button class="btn btn-primary d-none" id="loading_btn" type="button" disabled>
                        <span class="spinner-border spinner-border-sm" role="status" aria-hidden="true"></span>
                        Uploading...
                    </button>

                    <button type="button" id="cancel_btn" class="btn btn-secondary d-none">Cancel upload</button

                </div>

                <div id="progress_wrapper" class="d-none">
                    <label id="progress_status"></label>
                    <div class="progress mb-3">
                        <div id="progress" class="progress-bar" role="progressbar" aria-valuenow="25" aria-valuem
                    </div>
                </div>

                <div id="alert_wrapper"></div>

            </div>
        </div>
    </div>

    <!-- Import Bootstrap JavaScript here -->

</body>

</html>
```

If you're not using Flask, go ahead and replace `{{ request.url }}` with the URL of the endpoint you're posting to.

The JavaScript

```
// Get a reference to the progress bar, wrapper & status label
var progress = document.getElementById("progress");
var progress_wrapper = document.getElementById("progress_wrapper");
var progress_status = document.getElementById("progress_status");

// Get a reference to the 3 buttons
var upload_btn = document.getElementById("upload_btn");
var loading_btn = document.getElementById("loading_btn");
var cancel_btn = document.getElementById("cancel_btn");

// Get a reference to the alert wrapper
var alert_wrapper = document.getElementById("alert_wrapper");

// Get a reference to the file input element & input label
var input = document.getElementById("file_input");
var file_input_label = document.getElementById("file_input_label");

// Function to show alerts
function show_alert(message, alert) {

  alert_wrapper.innerHTML = `
    <div id="alert" class="alert alert-${alert} alert-dismissible fade show" role="alert">
      <span>${message}</span>
      <button type="button" class="close" data-dismiss="alert" aria-label="Close">
        <span aria-hidden="true">&times;</span>
      </button>
    </div>`

}

// Function to upload file
function upload(url) {

  // Reject if the file input is empty & throw alert
  if (!input.value) {

    show_alert("No file selected", "warning")

    return;

  }

  // Create a new FormData instance
  var data = new FormData();

  // Create a XMLHTTPRequest instance
  var request = new XMLHttpRequest();

  // Set the response type
  request.responseType = "json";

  // Clear any existing alerts
  alert_wrapper.innerHTML = "";

  // Disable the input during upload
  input.disabled = true;

  // Hide the upload button
  upload_btn.classList.add("d-none");

  // Show the loading button
  loading_btn.classList.remove("d-none");

  // Show the cancel button
  cancel_btn.classList.remove("d-none");

  // Show the progress bar
  progress_wrapper.classList.remove("d-none");

  // Get a reference to the file
  var file = input.files[0];

  // Get a reference to the filename
  var fileName = file.name;

  // Get a reference to the filesize & set a cookie
  var fileSize = file.size;
  document.cookie = `filesize=${fileSize};`;

  // Append the file to the FormData instance
  data.append("file", file);

  // request progress handler
  request.upload.addEventListener("progress", function (e) {

    // Get the loaded amount and total filesize (bytes)
    var loaded = e.loaded;
    var total = e.total;

    // Calculate percent uploaded
    var percent_complete = (loaded / total) * 100;

    // Update the progress text and progress bar
    progress.setAttribute("style", `width: ${Math.floor(percent_complete)}%`);
    progress_status.innerText = `${Math.floor(percent_complete)}% uploaded`;

  });

  // request load handler (transfer complete)
  request.addEventListener("load", function (e) {

    if (request.status == 200) {

      show_alert(`${request.response.message}`, "success");

    }
    else {

      show_alert(`Error uploading file`, "danger");

    }

    reset();

  });

  // request error handler
  request.addEventListener("error", function (e) {

    reset();

    show_alert(`Error uploading file`, "warning");

  });

  // request abort handler
  request.addEventListener("abort", function (e) {

    reset();

    show_alert(`Upload cancelled`, "primary");

  });

  // Open and send the request
  request.open("post", url);
  request.send(data);

  cancel_btn.addEventListener("click", function () {

    request.abort();

  })

}

// Function to update the input placeholder
function input_filename() {

  file_input_label.innerText = input.files[0].name;

}

// Function to reset the page
function reset() {

  // Clear the input
  input.value = null;

  // Hide the cancel button
  cancel_btn.classList.add("d-none");

  // Reset the input element
  input.disabled = false;

  // Show the upload button
  upload_btn.classList.remove("d-none");

  // Hide the loading button
  loading_btn.classList.add("d-none");

  // Hide the progress bar
  progress_wrapper.classList.add("d-none");

  // Reset the progress bar state
  progress.setAttribute("style", `width: 0%`);

  // Reset the input placeholder
  file_input_label.innerText = "Select file";

}
```

Last modified: 25 Feb 2019

Previous article
Dynamically increasing input textarea height with JavaScript

Next article
Infinite scrolling & lazy loading with JavaScript & Flask - IntersectionObserver

Related items

Julian Nash · 3 months ago in Flask
Application factory pattern | Learning Flask Ep. 30
Building scalable Flask applications from the start using the application factory pattern, blueprints and the current_app proxy
python flask Article Learning Flask

Julian Nash · 9 months ago in Python
MBAC - Method based access control
Using HTTP request methods as the primary means of access control
web Article

Julian Nash · 10 months ago in Python
Connecting to a Microsoft Azure Cosmos DB with Python and the MongoDB API
Debugging and connecting to an Azure Cosmos DB using the MongoDB API using PyMongo, PyMODM and MongoEngine
python mongo azure Article

Julian Nash · 10 months ago in Python
Pythons Enum module
Harnessing the power of Pythons enumerations and exploring the enum module
python Article

Contents
The route
The HTML
The JavaScript
Related items