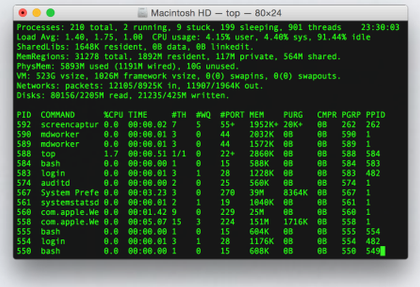
Storing Data and Files

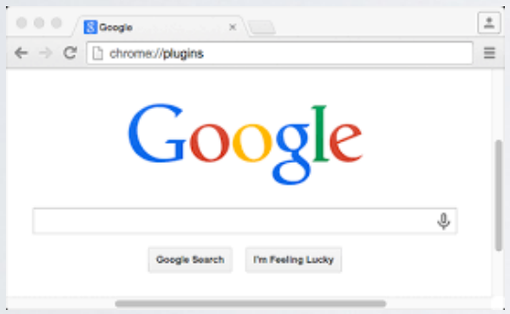
Thierry Sans

Storing Data in a Database

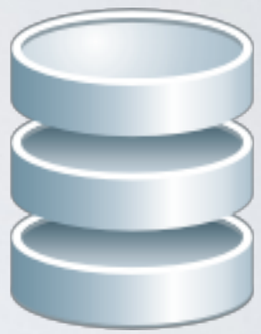
Modern Web Platform

Client Side





Server Side



Web API

Database

Why using a database

- Persistency
- Concurrency (avoid race conditions)
- Query
- Scalability

SQL vs NoSQL databases

Relational database (SQL database)

Data structure	tables and tuples
Query language	SQL
Inconvenient	not-optimized for big data analysis
Advantage	complex queries
Technology	*PostgreSQL, MySQL, MariaDB, SQLite, MSSQL*

NoSQL database

Data structure	key/value pairs
Query language	API style
Inconvenient	not adequate for complex queries
Advantage	optimized for big data analysis
Technology	*MongoDB, Redis, CouchDB, NeDB*

ORM - Object Relational Mapping

➡ Mapping between (OOP) objects and the database structure

Examples

- *Sequelize* for *PostgreSQL*, *MySQL*, *MariaDB*, *SQLite*
- *Mongoose* for *MongoDB*

Do/Don't

- Do **retrieve selected elements only**
  rather than retrieving an entire collection and filtering afterwards

- Do **define primary keys**
  rather than relying on auto-generated ones

- Do **split data into different collections**
  rather than storing list attributes

- Do **create join collections** whenever appropriate
  (only for NoSQL database without performant join feature)

Retrieving collections with paginated results

➡ Only retrieve what you need from a potentially large collection

Examples

```
GET /messages[?page=0]
GET /messages?page=1
GET /messages[?max=100]
GET /messages?max=20
```

Handling files

Browser restrictions

- It is **impossible** to write a piece of code that reads an arbitrary file in (client-side) Javascript

➡ Only files selected by users through file input forms can be processed

```
<form . . . >
      <input type="file" name="img" multiple>
      <input type="submit">
</form>
```

[optional] select multiple files

Sending a file from the terminal

```
$ curl -X POST
       -H "Content-Type: multipart/form-data"
       -F "picture=@localpath/to/img.png"
       -F "username=bart"
       http://...
```

Sending a file from the browser

- **Form action** (with page refresh)

```
<form action="/url"
      method="POST"
      enctype="multipart/form-data">
```

- **Fetch request** (without page refresh)

```
const file = document.get ...
const data = new FormData();
data("picture", file);
fetch( "/api/users/", {
  method: "POST",
     body: data
})
```

What is received on the server

File metadata

- filename
- mimetype (file type)
- size
- and others

File content

- Compressed binary or string

MIME types

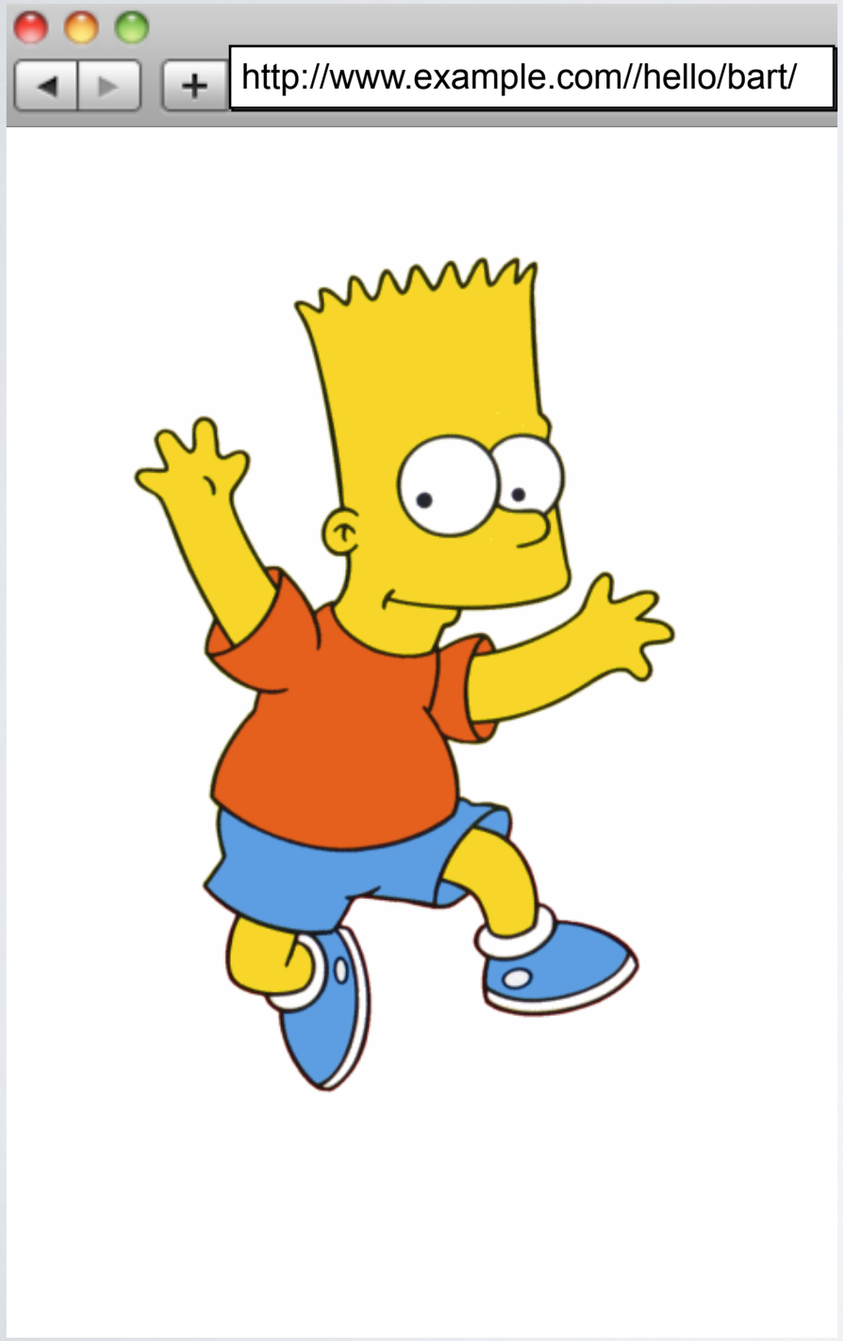
MIME (Multipurpose Internet Mail Extensions)
is also known as the **content type**

➡ Define the format of a document exchanged on internet (IETF standard) http://www.iana.org/assignments/media-types/index.html

Examples of MIME types

- text/html
- text/css
- text/javascript
- image/jpeg - image/gif - image/svg - image/png (and so on)
- application/pdf
- application/json

Example of how images are retrieved



GET hello/bart/

```
<html>
   <body>
      <img src=images/bart.jpg/>
   </body>
</html>
```

MIME : text/html

GET images/bart.jpg

MIME : image/jpg

Do/Don't with files

- Do **not** send a base64 encoded file content with JSON, use **`multipart/form-data`** instead (compression)
- Do **not** store uploaded files with the static content
- Do **not** serve uploaded files statically (security)
- Do store the mimetype and set the HTTP response header mimetype when files are sent back