Universidad del Rosario | Educación Continua

DESARROLLO DE APLICACIONES WEB

- Miguel Orjuela

Personas con *propósito* que ayudan a transformar vidas

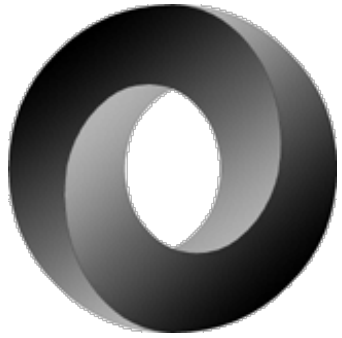
Sesión # 5

Intercambiando información con servidores Backend

Miguel Angel Orjuela Rocha

Ingeniero de Sistemas y Computación

JSON

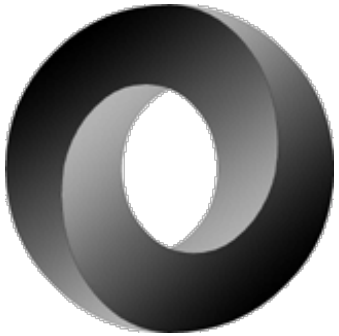


JSON - JavaScript Object Notation

Es un formato de texto ligero de intercambio de datos.

- Colección llave / valor
- Lista ordenada de valores

```
{
        nombre1 : valor1,
        nombre2 : valor2,
...
}
```

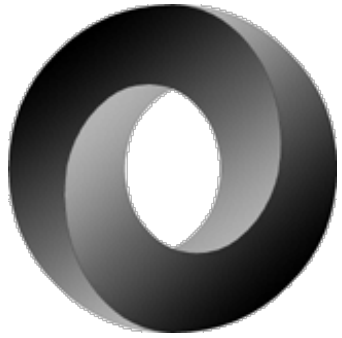


JSON - JavaScript Object Notation

Es usado con frecuencia para transferir datos entre un servidor y un navegador (cliente).

Un archivo .json puede estar dispuesto en una API o en archivos locales.

Ejemplo: https://jsonplaceholder.typicode.com/users



JSON - JavaScript Object Notation

Los valores de cualquier nombre JSON pueden ser strings, números, booleanos, nulos, matrices u objetos.

JSON es un **string** serializado que puede ser transformado a diferentes tipos de datos

[Objetivo] Acceder a JSON con Javascript

Usando JSON con Javascript

Usando datos de JSON con Javascript

[Paso 1] Creación de una variable “data” con un string JSON

```
var data = '[ { "name": "Aragorn", "race": "Human" }, { "name": "Gimli", "race": "Dwarf" } ]'
```

[Paso 2] Decodificar el string

```
data = JSON.parse(data);
```

Usando datos de JSON con Javascript

[Paso 3] Acceder a los datos como un objeto normal

```
console.log(data[1].name)

// Ejemplo: recorrer el objeto con iteraciones
for (var i = 0; i < data.length; i++) {
  console.log(data[i].name + ' is a ' + data[i].race + '.')
}
```

Acceder a un archivo JSON localmente

[Paso 1] Guardar los datos en un archivo llamado “data.json”

```
[
  {
    "name": "Aragorn",
    "race": "Human"
  },
  {
    "name": "Gimli",
    "race": "Dwarf"
  }
]
```

data.json

[Paso 2] Se instancia un nuevo objeto con el prototipo de un XMLHttpRequest()

```
var request = new XMLHttpRequest()
```

[Paso 3] Se abre la solicitud indicando que una solicitud GET se usará para abrir el archivo

```
request.open('GET', 'data.json', true)
```

[Paso 4] Especificarle a Javascript que hacer cuándo el archivo esté cargado

```
request.onload = function() {
        var data = JSON.parse(this.response) // this se refiere al      request en este contexto
        console.log(data) // Podemos ver los datos
        for( var i = 0; i < data.length; i ++){
           console.log(data[i].name + ' is a ' + data[i].race + '.') //      Se imprimen llaves y valores en la consola
  }
}
```

[Paso 5] Enviar la solicitud

```
request.send()
```

Conectarse a un API con Javascript

Objetivos

- Entender que es un API web
- Aprender a hacer solicitudes HTTP get con Javascript
- Crear y mostrar elementos HTML con Javascript

API -Application Program Interface

Son todos los **métodos de comunicación** entre varios componentes de software. Un API permite que un **software se comunique con otro software**.

API web

Son todos los **métodos de comunicación** que permiten que un **servidor** web interactúe con **software de terceros.**

Usa **solicitudes HTTP** para comunicarse para comunicarse con un **end point** de URL disponible públicamente y que contiene datos **JSON**

MODELO CRUD

	Descripción
CREAR => POST	Crear un nuevo recurso
LEER => GET	Recuperar un recurso
ACTUALIZAR => PUT//PATCH	Actualizar un recurso existente
ELIMINAR => DELETE	Eliminar un recurso

Paso 1

Crear un archivo de HTLM que contenga un enlace a un script y a un archivo de estilo

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,
initial-scale=1.0" />

  <title>Nueva App</title>

  <link
href="https://fonts.googleapis.com/css?family=Dosis:400,700"
rel="stylesheet" />
  <link href="style.css" rel="stylesheet" />
 </head>

 <body>
  <div id="root"></div>
  <script src="scripts.js"></script>
 </body>
</html>
```

index.html

Paso 2

Crear el archivo de estilos

```css
#root {
  max-width: 1200px;
  margin: 0 auto;
}

.container {
  display: flex;
  flex-wrap: wrap;
}

.card {
  margin: 1rem;
  border: 1px solid gray;
}

@media screen and (min-width: 600px) {
  .card {
    flex: 1 1 calc(50% - 2rem);
  }
}

@media screen and (min-width: 900px) {
  .card {
    flex: 1 1 calc(33% - 2rem);
  }
}
```

style.css

Paso 3

Conocer la API

https://ghibliapi.herokuapp.com/

Nombres de películas y descripciones

Esta API fue creada para ayudar a los desarrolladores a aprender cómo interactuar con los recursos mediante solicitudes HTTP

style.css

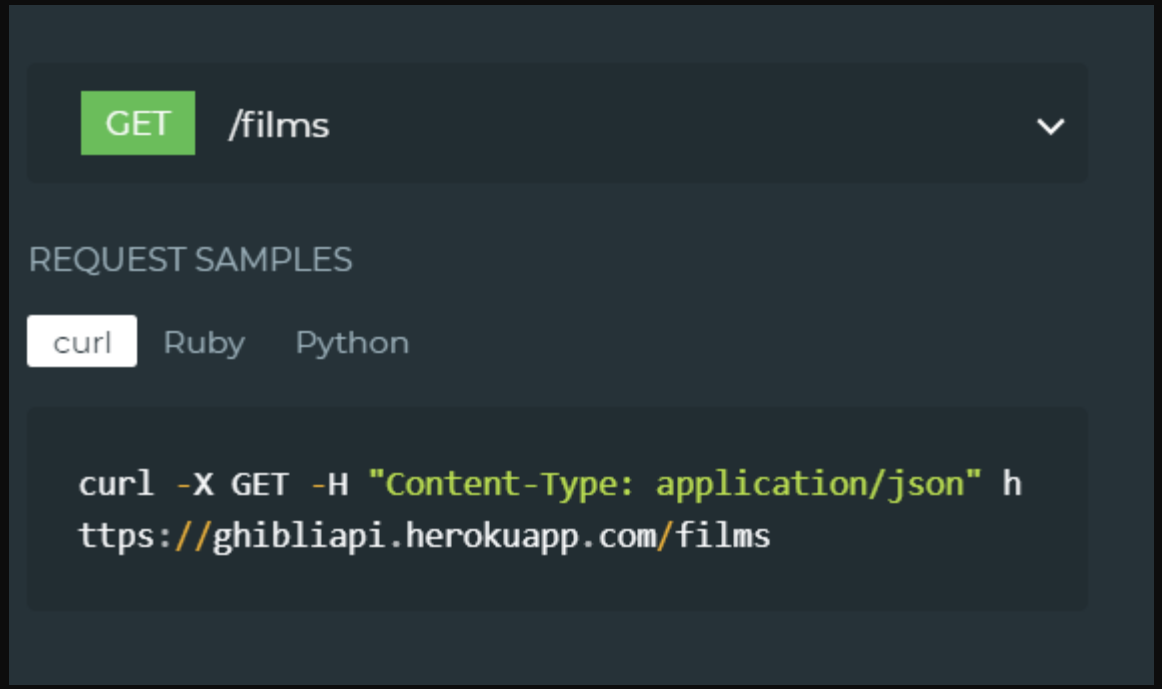
Paso 4

Obtener el end point de la API

Documentación:
https://ghibliapi.herokuapp.com/#tag/Films

API:

A la derecha es posible encontrar la solicitud “GET” JSON:
https://ghibliapi.herokuapp.com/films



style.css

Paso 5

Recuperar los datos con una solicitud HTTP en el archivo script.js

Ejercicio: Explique qué hace la solicitud

```
var request = new XMLHttpRequest()

request.open('GET',
'https://ghibliapi.herokuapp.com/films', true)

request.onload = function () {
    var data = JSON.parse(this.response)

    data.forEach(movie => {
        console.log(movie.title)
    })
}

request.send()
```

script.js

Paso 6

Modificar los elementos / crear elementos (Contenedor y logo)

```
const app = document.getElementById('root')

const logo = document.createElement('img')
logo.src = 'logo.png'

const container = document.createElement('div')
container.setAttribute('class', 'container')

app.appendChild(logo)
app.appendChild(container)
```

script.js

Paso 7

Modificar los elementos / crear elementos (Contenedor y logo)

```
const app = document.getElementById('root')

const logo = document.createElement('img')
logo.src = 'logo.png'

const container = document.createElement('div')
container.setAttribute('class', 'container')

app.appendChild(logo)
app.appendChild(container)
```

script.js

Paso 8

Almacenar los elementos de la petición

Dentro del código que ya se tenía desarrollado, reemplazarlo en el forEach por el código que crea cada contenedor de nombre y descripción de las películas

```
data.forEach(movie => {
  const card = document.createElement('div')
  card.setAttribute('class', 'card')

  const h1 = document.createElement('h1')
  h1.textContent = movie.title

  const p = document.createElement('p')
  movie.description = movie.description.substring(0, 300) // Limit to
300 chars
  p.textContent = `${movie.description}...`
  container.appendChild(card)

  card.appendChild(h1)
  card.appendChild(p)
})
```

script.js