Software Development for Web and Mobile

Pepe García

2020-04-20

Software Development for Web and Mobile

Plan for today

Today we will:

- Learn the basics of Javascript

- Understand how Javascript is use in web development

What's Javascript

Javascript is a dynamic programming language, multiparadign, and with weak typing.

What's Javascript

JS has become ubiquitous because it's the only web-native programming language.

Differences with Python

1. Indentation doesn't matter (although is better to indent your code). Blocks are delimited by curly brackets **{}**
2. functions are declared with **function**, not **def**.
3. variables are declared using **let**.
4. Convention to use **camelCase** instead of **under_score** for naming

JS is NOT Java

JS is **not** Java. The creators of JS decided to prefix it with Java as a marketing trick.

Not only in the browser

Although it initially was developed to be run on the browser, currently JS runs on several different platforms:

- Browser
- Natively (using **Node JS**)
- JVM (using **Rhino**)

- on Mobile phones (using **React native**)

Using JS

As with CSS there are several ways to include JS in a webpage

Using JS

We can use a <**script**> tag and inline the JS code inside.

See **inline-js.html**

Using JS

We can also include external JS files in our web page.

See **external-js.html**

Variables

Variables are created in JS using the **let** keyword:

```
let age = 28;
let name = "Pepe";
let lastName = "García";
```

Variables

Variables whose value never changes are called constants, and they're created with the ‘const‘ keyword:

```
const gravityAcceleration = 9.8;
gravityAcceleration = 33;
// Uncaught TypeError: Assignment to constant variable.
```

Functions

Functions are created in JS using the **function** keyword.

```
function <name> (<params>) {
    // do stuff
    return <return value>;
}
```

Functions

```
function areaTriangle(b, h) {
    return b * h / 2;
}
```

Arrow functions

```javascript
const areaTriangle = (b, h) => b * h / 2;
```

There's also a shorthand in Javascript for declaring **anonimous functions**, using **arrow functions**.

functional programming

Javascript is a very flexible programming language and allows us to use some functional programming.

In functional programming, we use functions as if it were values. Functions can be **taken as parameters**, **returned from other functions**, and **stored in variables**

functional programming

Create a function operate, that receives an operation and two numbers as parameters.

Then call this function with different numbers and operations.

see **operations.js**

Conditionals

As in Python, we use conditionals in JS to do different things in our program depending on a value.

Conditionals

```
if (<condition>) {
  // do stuff
} else if(<other condition>) {
  // do something else
} else {
  // to this otherwise
}
```

Boolean operators

Python	JS
==	===
!=	!===
and	&&
or	||
not	!

Arrays

arrays or **lists** are used to store collections of values in JS

```
let elements = [1,2,3];
elements[0] = 22;
let copyOfElements = elements.slice();
```

Array.push

We add an element to the end of an array using the **push method**

```
let elements = [1,2,3];
elements.push(4);
```

Array.pop

We remove an element in the given position of an array with the pop method.

```
let elements = [1,2,3];
elements.pop(0);
```

Objects

Objects are key-value pairs. We create them using curly brackets:

```
let beatles = {
  drummer: "Ringo",
  guitarist: "George",
  bassist: "Paul",
  singer: "John"
}
```

Objects

We can access the values of the object as if they were **properties** or using the **key**:

```
beatles["drummer"]


beatles.drummer
```

Loops

As in Python, we can loop using **while** and **for** loops.

While loops

```
while(<condition>) {
  <body>
}
```

For loops

The for loop is kind of different from the one in Python.

It receives some config, in which we specify three different sections separated by semicolons (**;**):

1. The creation of the *loop variable*. It can be something like **let i = 0**.
2. The condition that needs to be truthy for the loop to keep iterating. **i < 33.**
3. The update we do to the *loop variable* on every iteration. **i++**.

For loops

One can also use loops in a way similar to Python using the shorthand syntax:

```
for (let value in elements) {
  <body>;
}
```

For loops

Simple looping.

Iterate over all elements of an array

see **arrays.js**

Exercise

Create a function to check if a given array is *palindromic.*

Keep in mind that you'll need to implement a function to check if arrays are equal.

The DOM

The DOM (**Document Object Model**) is the representation of the HTML of a webpage that we have available in Javascript. We can modify/access/create/delete HTML elements directly from Javascript, and we do it using the DOM.

The DOM

The DOM (**Document Object Model**) is the representation of the HTML of a webpage that we have available in Javascript. We can modify/access/create/delete HTML elements directly from Javascript, and we do it using the DOM.

The DOM

```
document
    .querySelector("h1")
    .innerHTML = "potato";
```

document points to the root of the DOM

querySelector is a method that we use to obtain the first element that matches a CSS selector

innerHTML is the attribute that represents the HTML inside an element

changing inner HTML

The DOM

```
document
    .querySelector("h1")
    .classList.add("my-class");
```

classList is an attribute of HTML elements with which we can manage its classes

Adding classes

The DOM

```
const parent = document
    .querySelector("div");

const child = document
    .createElement("div");

child.innerText = "this is the inner text";

parent.appendChild(child);
```

creating new elements

We can create new elements with **document.createElement**
We can add them later to other elements with **parent.appendChild(child)**

Exercises

```
const bands = [
    {
      name: "The Beatles",
      instruments: {
        John: "voice",
        Paul: "bass",
        Ringo: "drums",
        George: "guitar"
      }
    },
    {
      name: "The Ramones",
      instruments: {
        Johnny: "voice",
        Joey: "guitar",
        Marky: "drums",
```

Events

Events are at the very heart of JS. Some even say that it's an event oriented language. With events we can handle how a webpage reacts to certain actions.

Examples of events

- click in a button
- scroll
- change the contents of a text field
- a timer expires
- data from the server arrives

Handling events

When handling events in JS we'll need to:

- select the element
- add the handler function
- add the event listener

Handling events

```
// select the element
const button = document.querySelector('.button-clicky');

// create a handler
const showAlert = () => console.log('button clicked!');

// add the listener
button.addEventListener('click', showAlert);
```

Handling events

see **events.js**

Exercises

Add four buttons to your previous web page, one saying voice, other saying bass, other saying drums, and other saying guitar.

Make sure that, when a button is clicked, the member that plays the given instrument in all bands gets highlighted.

references

https://books.adalab.es/materiales-front-end-e

Homework

White belt

Create a simple webpage in which, when a button is clicked, all the links change their background to blue and their text color to white.

Blue belt

Investigate the functional methods on array. Namely **map**, **filter**, **forEach**, and **reduce**.

Try to apply them to the following cases:

- given an array of numbers, return only the **even ones**
- given an array of numbers, return its **sum**
- given an array of numbers, **log all** in the console
- given an array of numbers, return a new array with **all elements squared**

Black belt

Investigate about forms in HTML.

Create a **simple** web page in which the user can write the name of a song in an **input** field and get the lyrics of that song.

You'll also need to investigate how to do HTTP requests from Javascript (https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch).

This is the API you'll need to use https://lyricsovh.docs.apiary.io/#reference/0/lyrics-of-a-song/search?console=1