**Week 03**

**UX – User Experience**

Many people users the smartphone today and the number of them only increase, with mobile them being able to have access the information and features anywhere and anytime, for this offer the good experiences to users is essential to get news users and make them to have engaged in your site/app. It's not just about good design is necessary more than this, need to provide users with the ability to learn simple tasks, as well guarantee efficient in your functionalities and elements to help the users to memorize how use the system/site/app.

Some principles help us in this challenge: Content Prioritization, Make Navigation Intuitive, Touchscreen Target Sizes, Provide User Control, Legible Text Content, Make Interface Elements Clearly Visible, Hand Position Controls, Minimize Data Input, Create a Seamless Experience and Test Your Design.

# **APIs**

Application Programming Interfaces (APIs) are constructs made available in programming languages to allow developers to create complex functionality more easily. They abstract more complex code away from you, providing some easier syntax to use in its place.

Normally have two API's categories: Browsers APIs or Third-party APIs. Browsers APIs are built into your web browser and can expose data from the browser and surrounding computer environment and do useful complex things with it already Third-part APIs are not built into the browser by default, and you generally have to retrieve their code and information from somewhere on the Web.

**Browsers APIs**

APIs for manipulating documents loaded into the browser, example DOM (Document Object Model);

APIs that fetch data from the server to update small sections of a webpage on their own are very commonly used, for example *XMLHttpRequest* and *Fetch API* the *Ajax* term describe this technique.

APIs for drawing and manipulating graphics are widely supported in browsers — the most popular ones are *Canvas* and *WebGL*, which allow you to programmatically update the pixel data contained in an *HTML <canvas> element* to create 2D and 3D scenes, such APIs are often combined with APIs for creating animation loops example *window.requestAnimationFrame()* *and others* to make constantly updating scenes like cartoons and games.

[Audio and Video APIs](https://developer.mozilla.org/en-US/docs/Web/Guide/Audio_and_video_delivery) like [HTMLMediaElement](https://developer.mozilla.org/en-US/docs/Web/API/HTMLMediaElement), the [Web Audio API](https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API), and [WebRTC](https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API) allow you to do things with multimedia.

Device APIs enable you to interact with device hardware: for example, accessing the device GPS to find the user's position using the[*Geolocation API*](https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API)*.*

Client-side storage APIs allow to store client information.

*Asynchronous operation is when you invoke the function only when the operation has completed*

*A function that is taken by another function as an argument is called a callback function. A callback function is a function passed to another function as an argument, which is then invoked within the outer function to complete some kind of routine or action.*

*A Promise is a proxy for a value not necessarily known when the promise is created. It allows you to associate handlers with the success value or reason for failure of an asynchronous action.*

**How do APIs work?**

Your code interacts with APIs using one or more JavaScript objects, which serve as containers for the data the API uses (contained in object properties), and the functionality the API makes available (contained in object methods).

**setTimeout and setInterval**

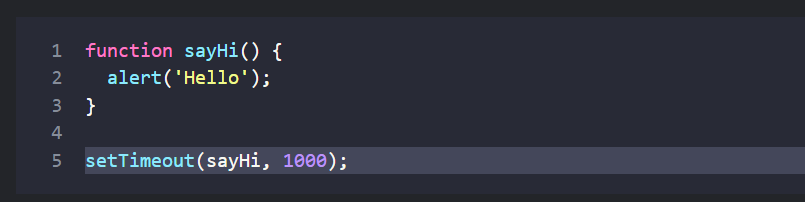
setTimeout and setInterval allow registering one function to be called after one quantity time, they are global functions client-side, and they are definite as how window methods.

The method setTimeout() executes one function after elapsing one number of specified milliseconds and returns one value that can be passed to clearTimeout() to cancel the execution.

The method setInterval() is like setTimeout() except that the function is called repeatedly in intervals the number of milliseconds specified, and your value can be passed to clearInterval() to cancel any call future the function.

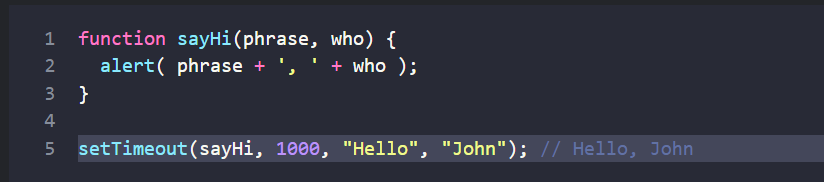
## **setTimeout**

The syntax:



The delay before run, in milliseconds (1000 ms = 1 second), by default 0.

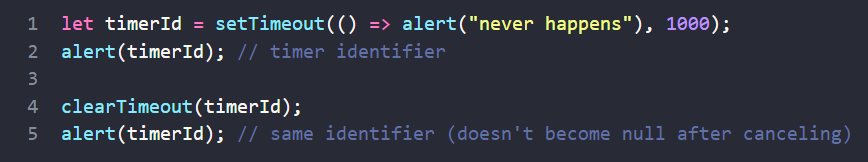
With arguments:



If the first argument is a string, then JavaScript creates a function from it.

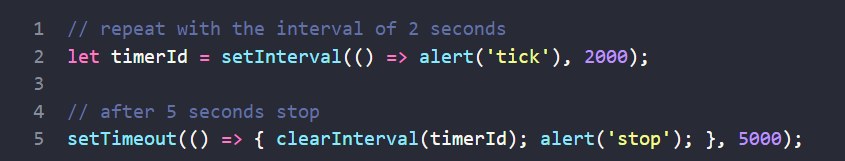
### **Canceling with clearTimeout**

The syntax to cancel:



## **SetInterval**

The setInterval method has the same syntax as setTimeout.



There’s a special use case: setTimeout(func, 0), or just setTimeout(func). This schedules the execution of func as soon as possible. But the scheduler will invoke it only after the currently executing script is complete.