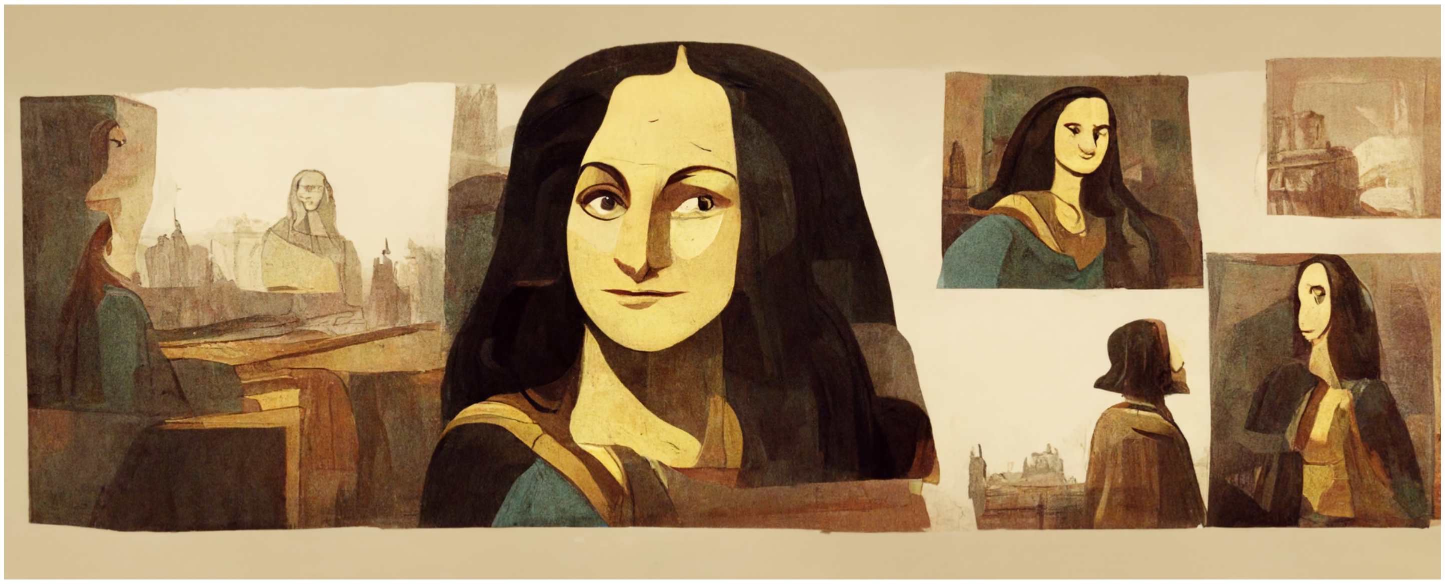
9.1 Async Timing



mona lisa - answered by April Joy Ipac

- JavaScript allows for non-blocking execution of code by delaying the execution of specific tasks (using timeouts) or repeatedly executing tasks at set intervals (using intervals) without halting the main program flow, enabling responsive and concurrent operations.

Keyword	Description	Arguments	Sample code	Code explanat

`setTimeout(function, timer)`	• Schedules the execution of a specified function or code block after a specified time delay	• `function` (required) - the name of the function that would be ran
		• `timer` (required) the number of milliseconds to wait before the function is run.
			`<div id="fruit-container">`
			`<p id="fruit-name"></p>`
			`</div>`
			
			`const fruits = [`
			`"Apple",`
			`"Banana",`
			`"Orange",`
			`"Grape",`
			`];`
			
			`<script>`
			`let current_index = 0;`
			
			`function displayFruit() {`
				• This coc continuc cycles through of fruits, displayi each on paragra element a 2-seco interval betweer `changeR` the func `display` which

			`  const fruit_name = $("#fruit-name");`
			`  const current_fruit = fruits[current_index];`
			
			`  fruit_name.text(current_fruit);`
			`  current_index = (current_index + 1) % fruits.length;`
			
			`  setTimeout(displayFruit, 2000);`
			`}`
			
			`displayFruit();`
			`</script>`
				updates content paragra element the nam the curr fruit and `setTime` call itsel 2 secon

`setInterval(function, timer)`	• Repeatedly executes a specified function or code block at fixed time intervals, creating a form of asynchronous, repetitive behavior in your code.		`<div id="fruit-container">`
			`<p id="fruit-name"></p>`
			`</div>`
			
			`<script>`
			`const fruits = [`
			`  "Apple",`
			`  "Banana",`
			`  "Orange",`
			`  "Grape",`
			`];`
			
			`let current_index = 0;`
			
			`function displayFruit() {`
			`  const fruit_name = $("#fruit-name");`
			`  const current_fruit = fruits[current_index];`
			`  fruit_name.text(current_fruit);`
			`  current_index = (current_index + 1) %`
				• This coc updates content paragra element the nam the curr fruit and iterates through fruits arr
				• It uses `setInte` to repea call the `display` function 2 secon resulting continu rotation displaye fruits.

			`fruits.length;`
			`}`
			
			`setInterval(displayFruit, 2000);`
			`</script>`	

Waiting for a Timeout

- When using the JavaScript function `setTimeout()`, you can specify a callback function to be executed on time-out.

```
setTimeout(myFunction, 3000);

function myFunction() {
  document.getElementById("demo").innerHTML = "I love You !!";
}
```

- In the example above, `myFunction` is used as a callback.
- `myFunction` is passed to `setTimeout()` as an argument.
- 3000 is the number of milliseconds before time-out, so myFunction() will be called after 3 seconds.
- Instead of passing the name of a function as an argument to another function, you can always pass a whole function instead:

```
setTimeout(function() { myFunction("I love You !!!"); }, 3000);

function myFunction(value) {
  document.getElementById("demo").innerHTML = value;
}
```

- In the example above, `function(){ myFunction("I love You !!!"); }` is used as a callback. It is a complete function. The complete function is passed to `setTimeout()` as an argument.
- 3000 is the number of milliseconds before time-out, so `myFunction()` will be called after 3 seconds.

Waiting for Intervals

- When using the JavaScript function `setInterval()`, you can specify a callback function to be executed for each interval:

```
setInterval(myFunction, 1000);

function myFunction() {
  let d = new Date();
  document.getElementById("demo").innerHTML=
  d.getHours() + ":" +
  d.getMinutes() + ":" +
  d.getSeconds();
}
```

- In the example above, `myFunction` is used as a callback.
- `myFunction` is passed to `setInterval()` as an argument.
- 1000 is the number of milliseconds between intervals, so `myFunction()` will be called every second.

Additional Material

- **Learn more**
  - W3Schools