Client-side Technologies

Dr. Niveen Nasr El-Den
iTi

Day 6

JavaScript Fundamentals cont.

Browser Object Model

BOM

Browser Engine & JavaScript

- **Browser engine** is a core software component of every major web browser. The primary job of a browser engine is to transform HTML documents and other resources of a web page into an interactive visual representation on a user's device.
  - e.g. Blink, Gecko, webkit etc.

- All Chromium-based browsers use Blink browser engine.
- **JavaScript engine** is a computer program that executes JavaScript (JS) code
  - e.g. V8, spiderMonkey etc.
- In 2019, Microsoft announced plans to rebuild the browser as Chromium-based with Blink and V8 engines.

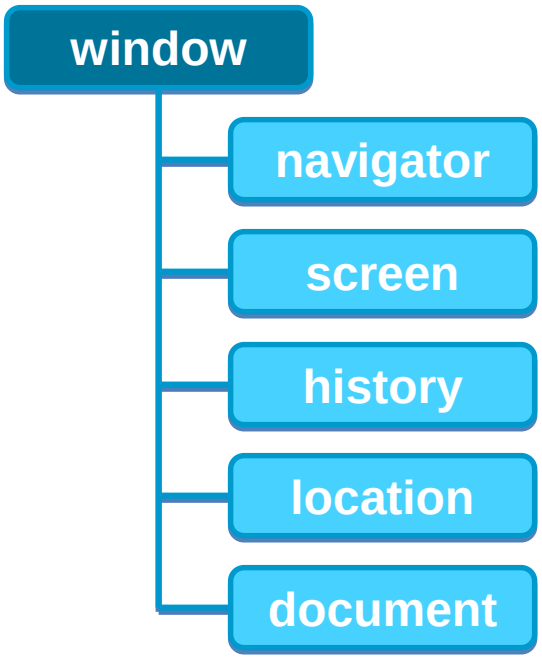
https://en.wikipedia.org/wiki/Browser_engine

BOM

- BOM Stands for Browser Object Model.
- BOM covers objects which relate to the browser.
- At the top of the **BOM** hierarchy is **window** object. Below that comes the
  - **navigator** object,
  - **screen** object,
  - **history** object,
  - **location** object, and
  - **document** object
    - It is the top level of the **DOM** hierarchy.

  Each object below the window is of equal status. (comes in no particular order).

  They all relate directly to the window object.



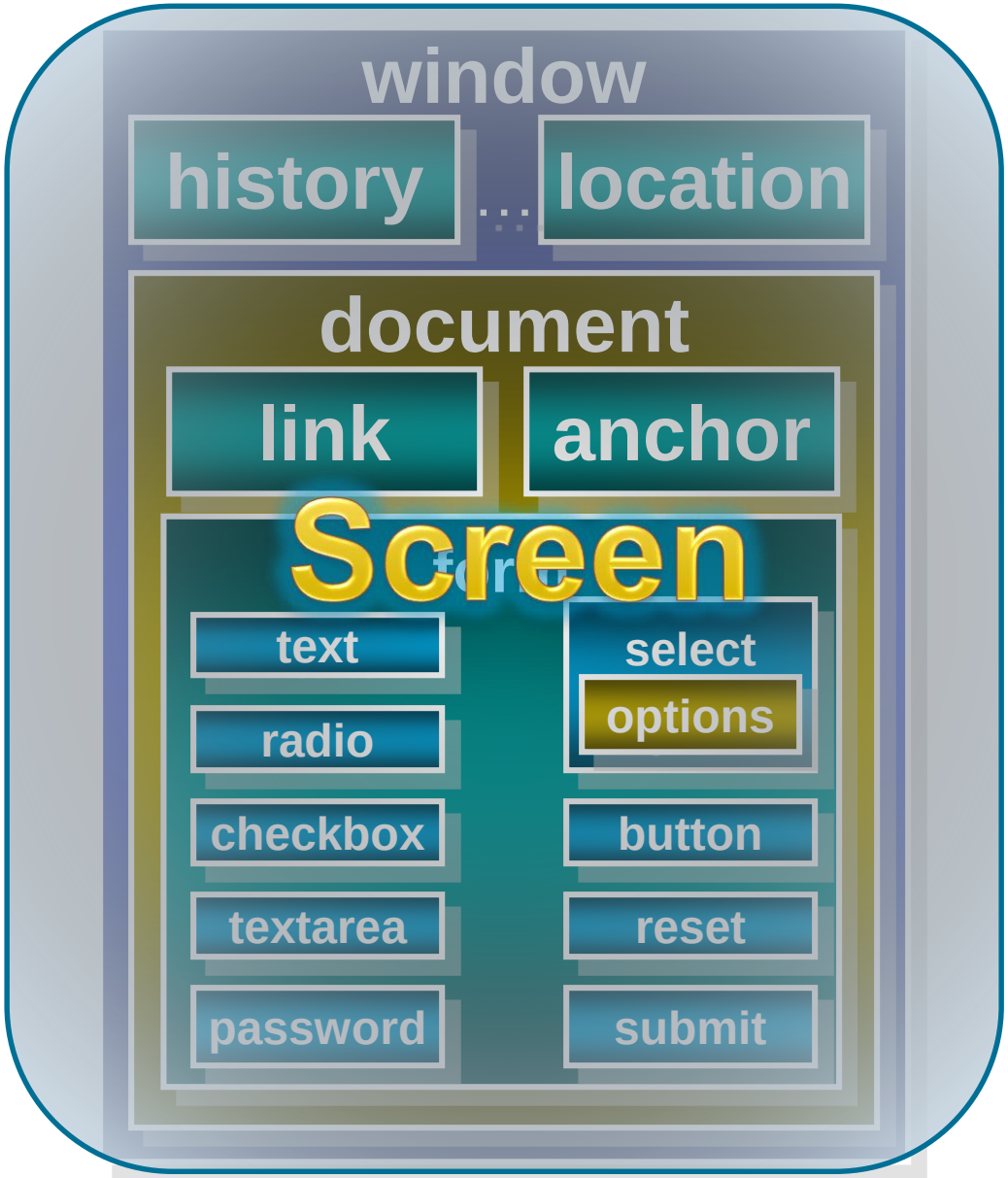
BOM

- Using the **BOM**, developers can move the window, and perform other actions that do not directly relate to the page content.
- For some reason, the **B**rowser **O**bject **M**odel is generally not referred to by its proper name. More often, it's usually wrapped up with the **DOM**.
- In actuality, the **DOM**, which relates to all things pertaining to the document, resides *within* the **BOM**.
- Because no standards exist for the BOM, each browser has its own implementation.

JavaScript Top Object Model Hierarchy

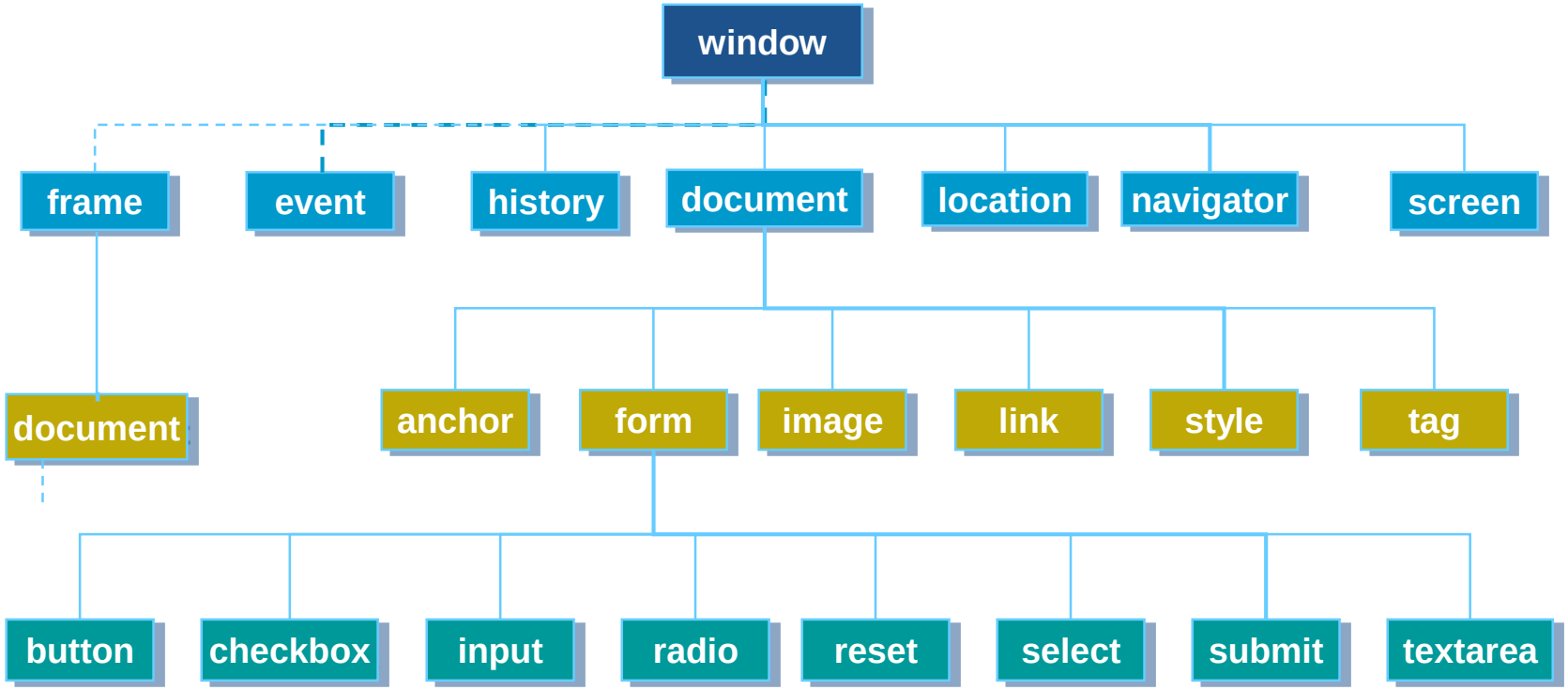
- Every page has the following objects:
  - **window** *:* the top-level object; has properties that apply to the entire window.
  - **navigator** *:* has properties related to the name and version of the Navigator being used.
  - **document** *:* contains properties based on the content of the document, such as title, background color, links, and forms.
  - **location** *:* has properties based on the current URL.
  - **history** *:* contains properties representing URLs the client has previously requested.
  - **screen** : contains information about the visitor's screen.

Browser Model



Model Hierarchy

BOM is a larger representation of everything provided by the browser including any other functionality the browser may expose to JavaScript.



Window

- Window is the top level object in the JavaScript client hierarchy.
- Window is the Global Object
- The Window object represents a browser window.
- Window object has a set of properties & methods.
- Object Model Reference:

  window
- To reference its properties & methods:
  - [window.]property
  - [window.]method

Window Properties

Name	Description	Syntax
document	Reference to the current document object.	window.document
frames	An array referencing all of the frames in the current window.	window.frames[i]
history	Reference to the History object of JavaScript	window.history
navigator	Reference to the browser application	window.navigator
location	Reference to the Location object of JavaScript	window.location

Window Methods

Name	Description	Syntax
alert()	Displays an alert box with a message and an OK button	window.alert(“Hello”)
confirm()	Displays a dialog box with a message and an OK, returning true, and a Cancel, returning false	Window.confrim(“Do you want to exit”)
prompt()	Displays a dialog box that prompts the user for input	name=prompt("Please enter your name","")
open()	Opens a new browser window	window.open(URL, name [, features])
close()	close a specified window	window.close()
blur()	Sets focus away from the window.	window.blur()
focus()	Set calling window object on top	window.focus()
print()	Print the contents of the specified window.	window.print()

Window Properties

Name	Description	Syntax
document	Reference to the current document object.	window.document
frames	An array referencing all of the frames in the current window.	window.frames[i]
history	Reference to the History object of JavaScript	window.history
navigator	Reference to the browser application	window.navigator
location	Reference to the Location object of JavaScript	window.location

Window Methods

Name	Description	Syntax
alert()	Displays an alert box with a message and an OK button	window.alert(“Hello”)
confirm()	Displays a dialog box with a message and an OK, returning true, and a Cancel, returning false	Window.confrim(“Do you want to exit”)
prompt()	Displays a dialog box that prompts the user for input	name=prompt("Please enter your name","")
open()	Opens a new browser window	window.open(URL, name [, features])
close()	close a specified window	window.close()
blur()	Sets focus away from the window.	window.blur()
focus()	Set calling window object on top	window.focus()
print()	Print the contents of the specified window.	window.print()

Window Methods

Name	Description	Syntax
moveTo(h,v)	Moves the window to horizontal and vertical position relative top-left of screen	window. moveTo(,)
moveBy(h,v)	Moves the window by + or - horizontal and vertical pixels	window.moveBy(,)
resizeTo(h,v)	Changes the size of the window to horizontal and vertical number of pixels	window.resizeTo(,)
resizeBy(h,v)	Changes the size of the window by + or - horizontal and vertical pixels	window.resizeBy(,)
scrollTo(h,v)	Scrolls the document in the current window or frame to horizontal and vertical pixel postions from top of document	window.scrollTo(,)
scrollBy(h,v)	Scrolls the document in the current window or frame by + or - horizontal and vertical pixel from current position	window.scrollBy(,)

Window Methods (WindowTimers)

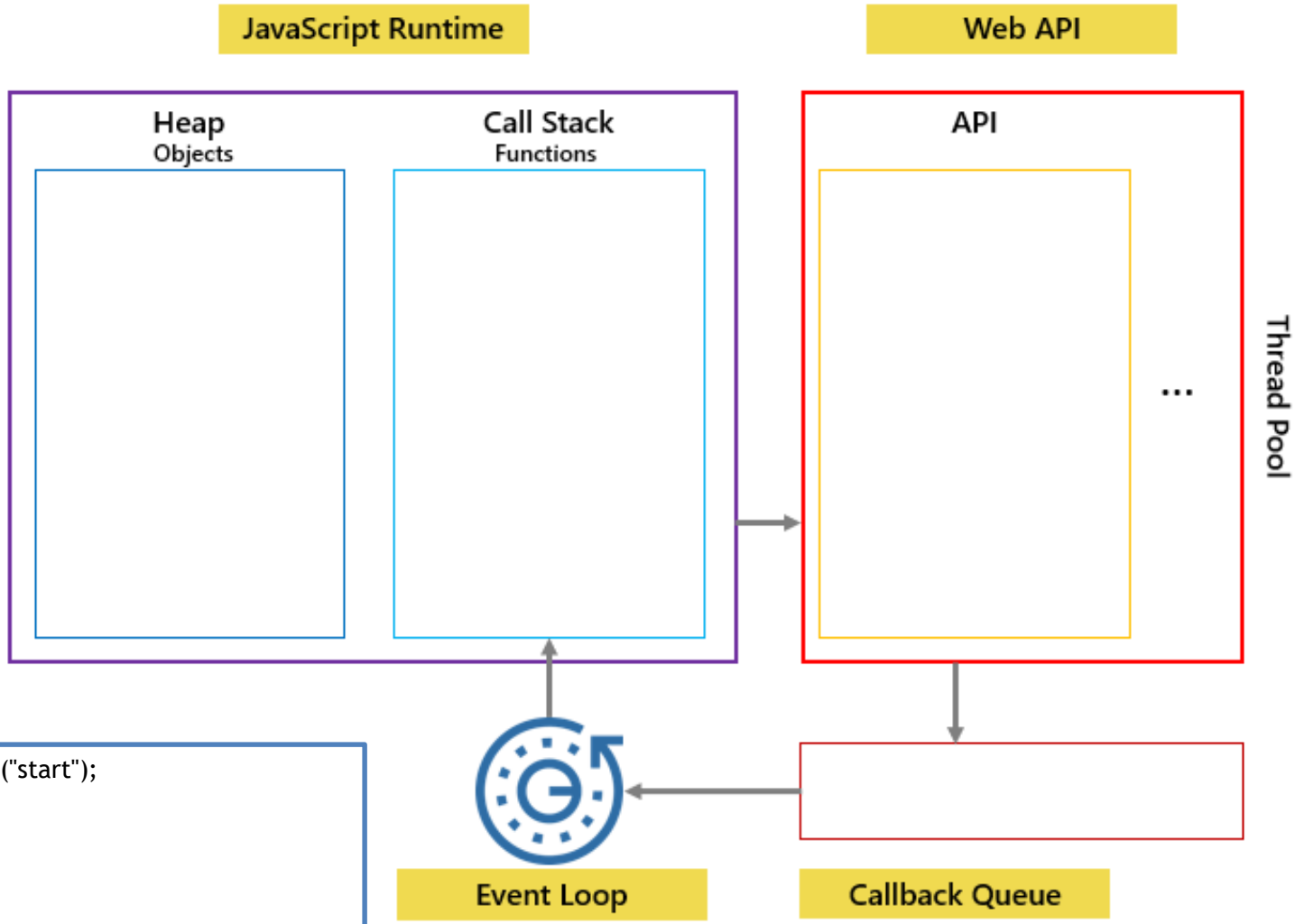
Name	Description	Syntax
setInterval()	Evaluates an expression at specified intervals	window.setInterval(*exp*, *time_interval*)
clearInterval()	Used to clear a time interval set using the above method	clearInterval(id_of_setInterval)
setTimeout()	Evaluates an expression after a specified number of milliseconds	window.setTimeOut(*exp*, *time_interval*)
clearTimeout()	Used to clear a timeout set using the above method	clearTimeout(id_of_setTimeout)

Example!

Example

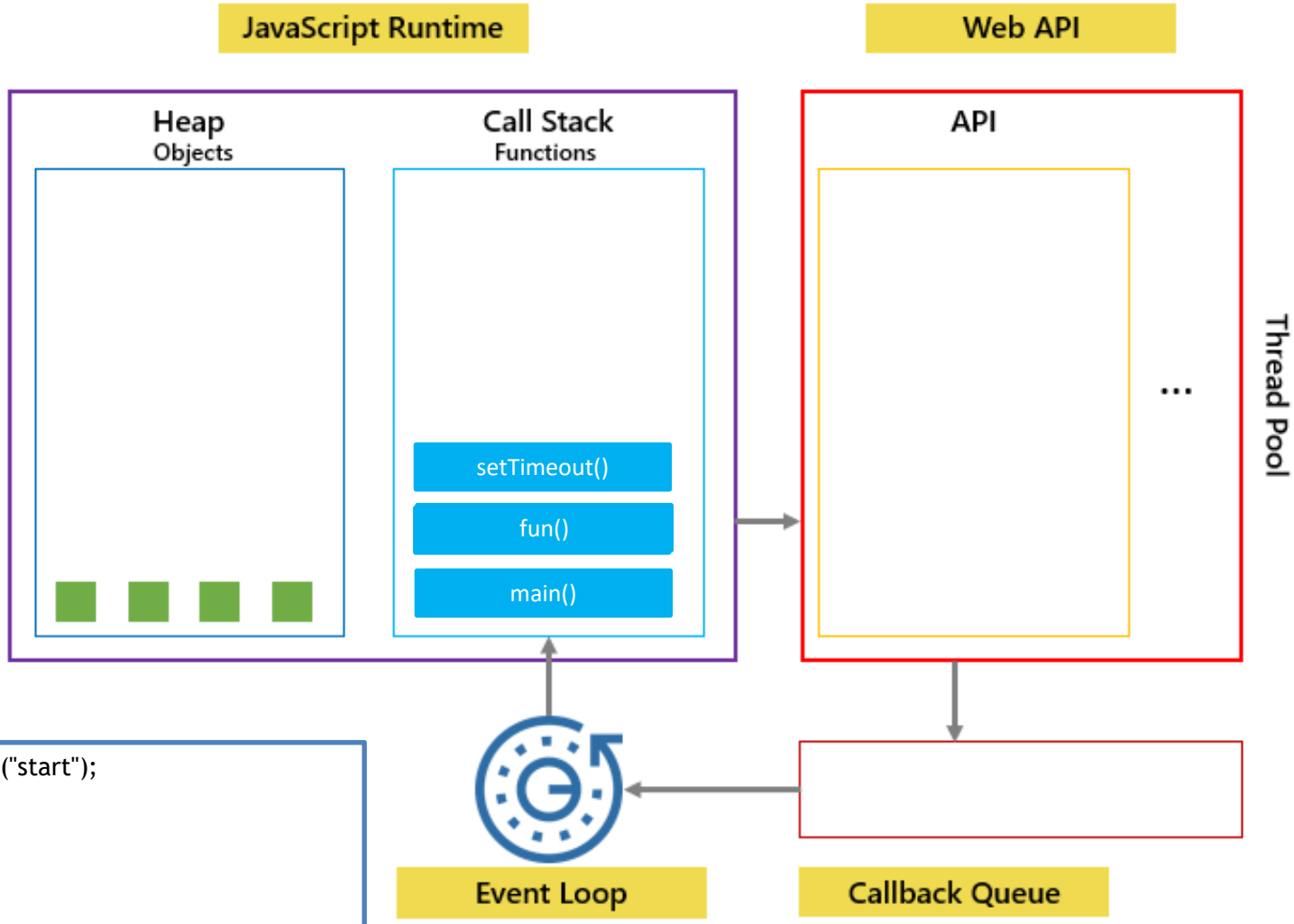
```
console.log("start");

wait5sec();

fun();

console.log("end");

setTimeout(function () {
    console.log("timeout after 1 sec");
}, 1000)
```

```
function fun() {
    setTimeout(function () {
        console.log("timeout immediately");
    }, 0);
}
```



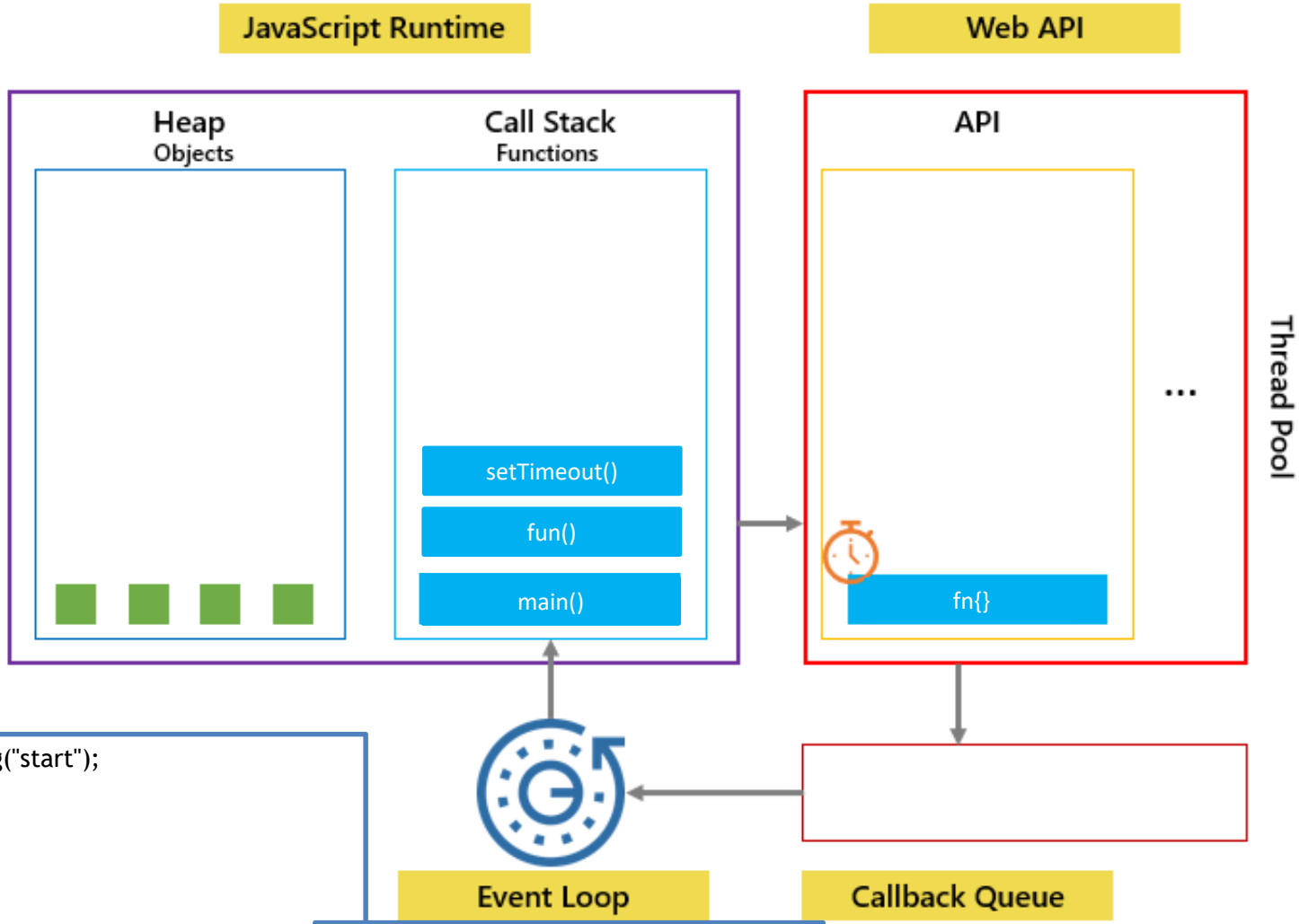

```
console.log("start");

wait5sec();

fun();

console.log("end");

setTimeout(function () {
    console.log("timeout after 1 sec");
}, 1000)
```

```
function fun() {
    setTimeout(function () {
        console.log("timeout immediately");
    }, 0);
}
```



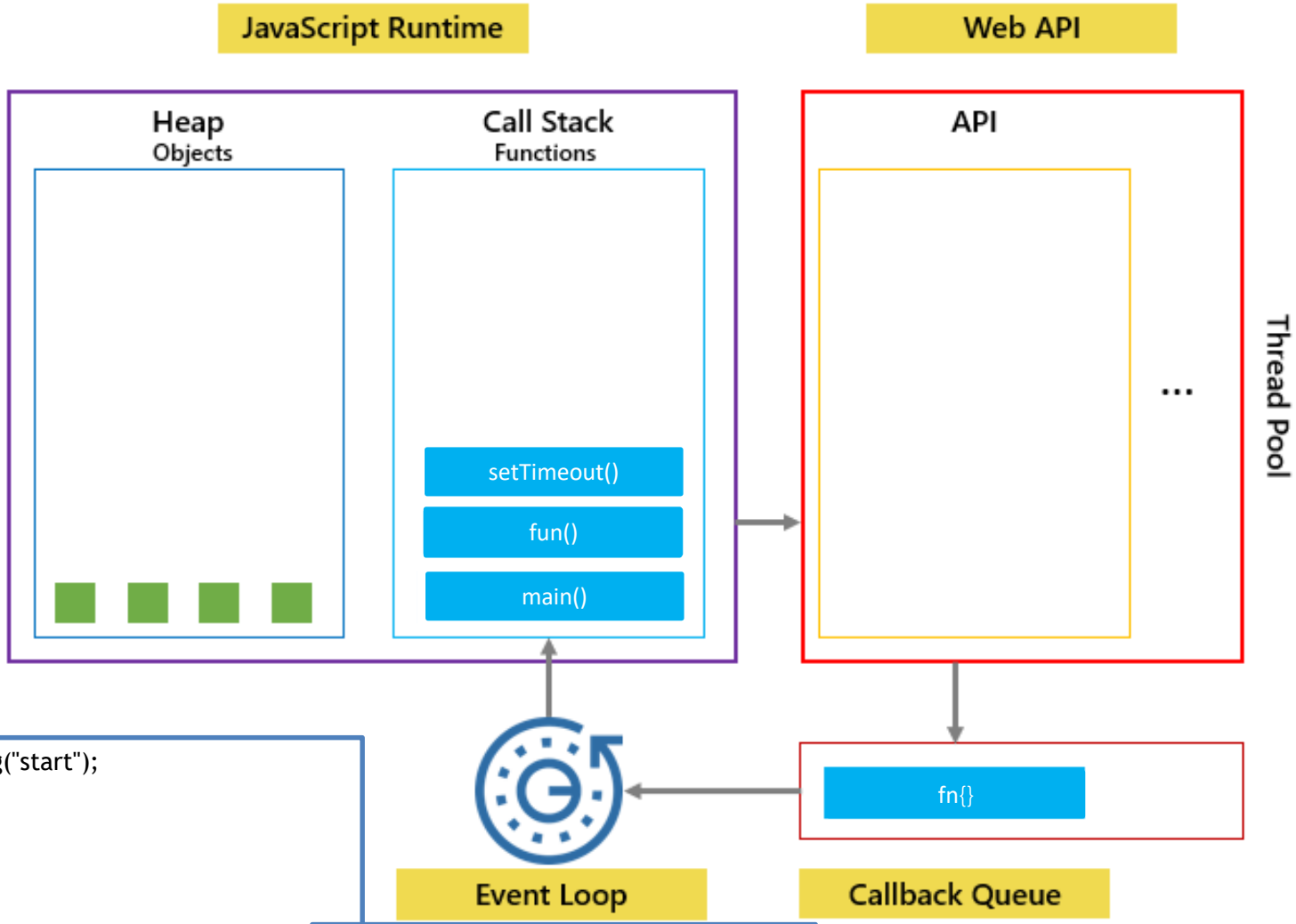

```
console.log("start");

wait5sec();

fun();

console.log("end");

setTimeout(function () {
    console.log("timeout after 1 sec");
}, 1000)
```

```
function fun() {
    setTimeout(function () {
        console.log("timeout immediately");
    }, 0);
}
```



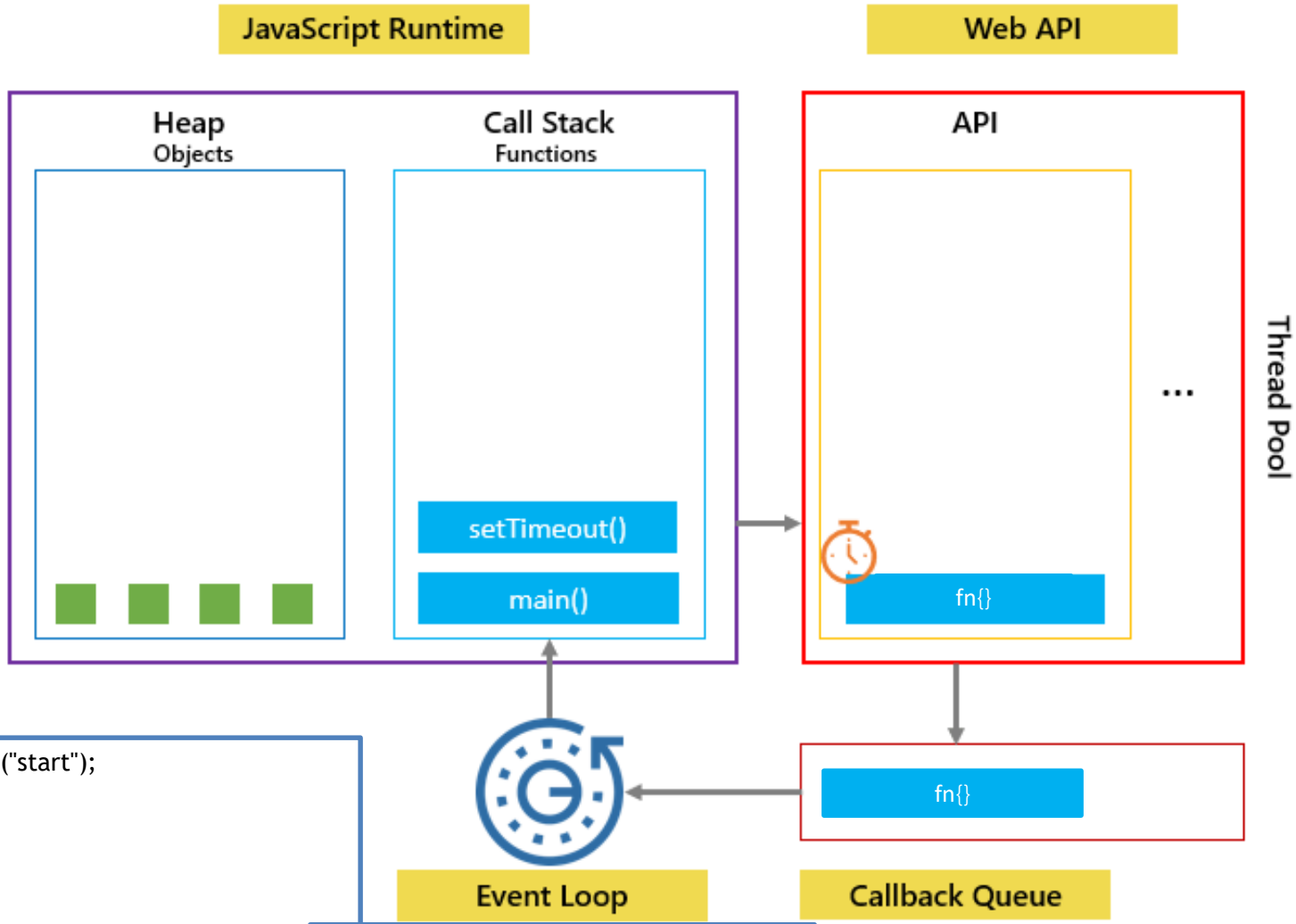

```
console.log("start");

wait5sec();

fun();

console.log("end");

setTimeout(function () {
    console.log("timeout after 1 sec");
}, 1000)
```

```
function fun() {
    setTimeout(function () {
        console.log("timeout immediately");
    }, 0);
}
```



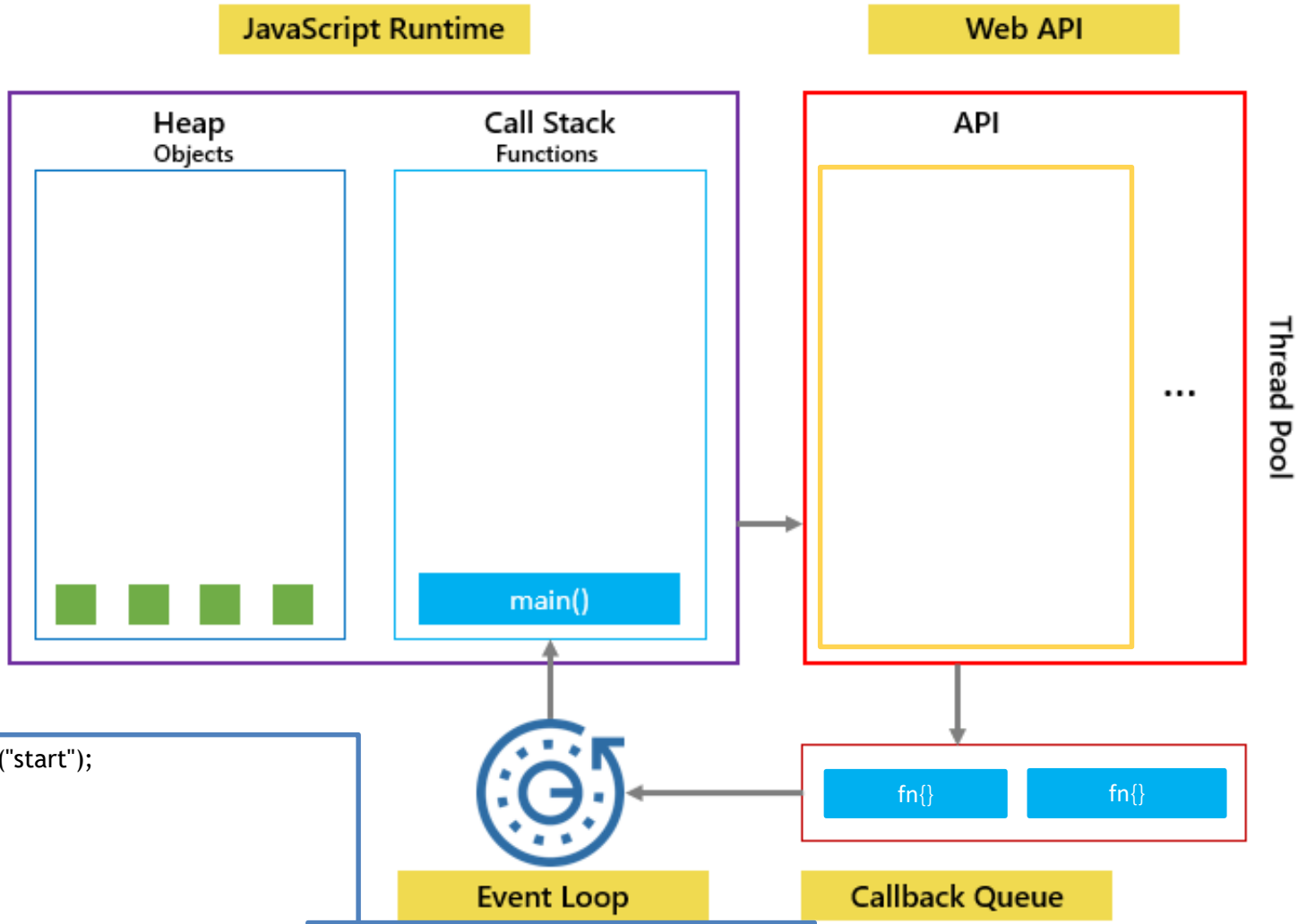

```
console.log("start");

wait5sec();

fun();

console.log("end");

setTimeout(function () {
    console.log("timeout after 1 sec");
}, 1000)
```

```
function fun() {
    setTimeout(function () {
        console.log("timeout immediately");
    }, 0);
}
```



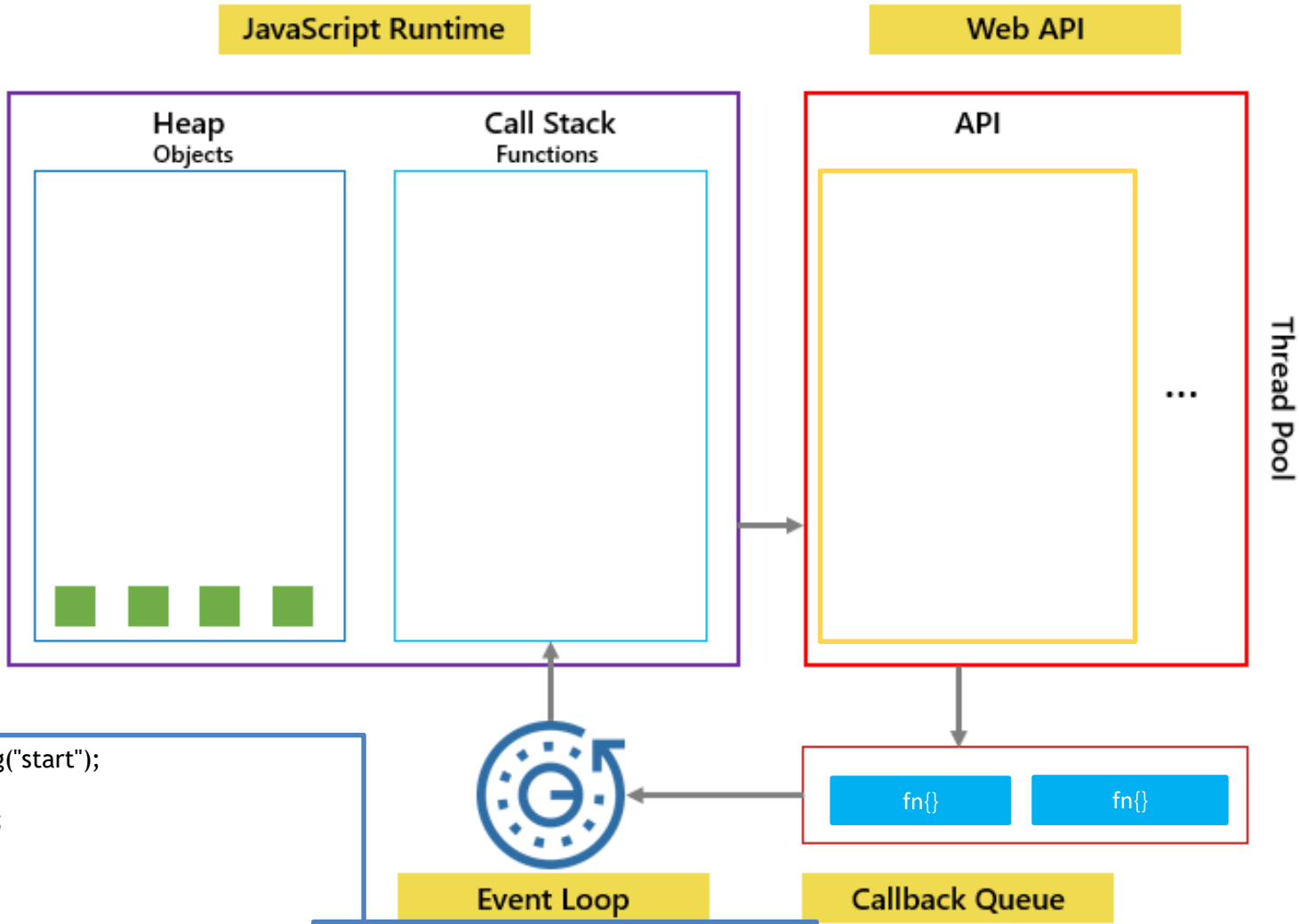

```
console.log("start");

wait5sec();

fun();

console.log("end");

setTimeout(function () {
    console.log("timeout after 1 sec");
}, 1000)
```

```
function fun() {
    setTimeout(function () {
        console.log("timeout immediately");
    }, 0);
}
```



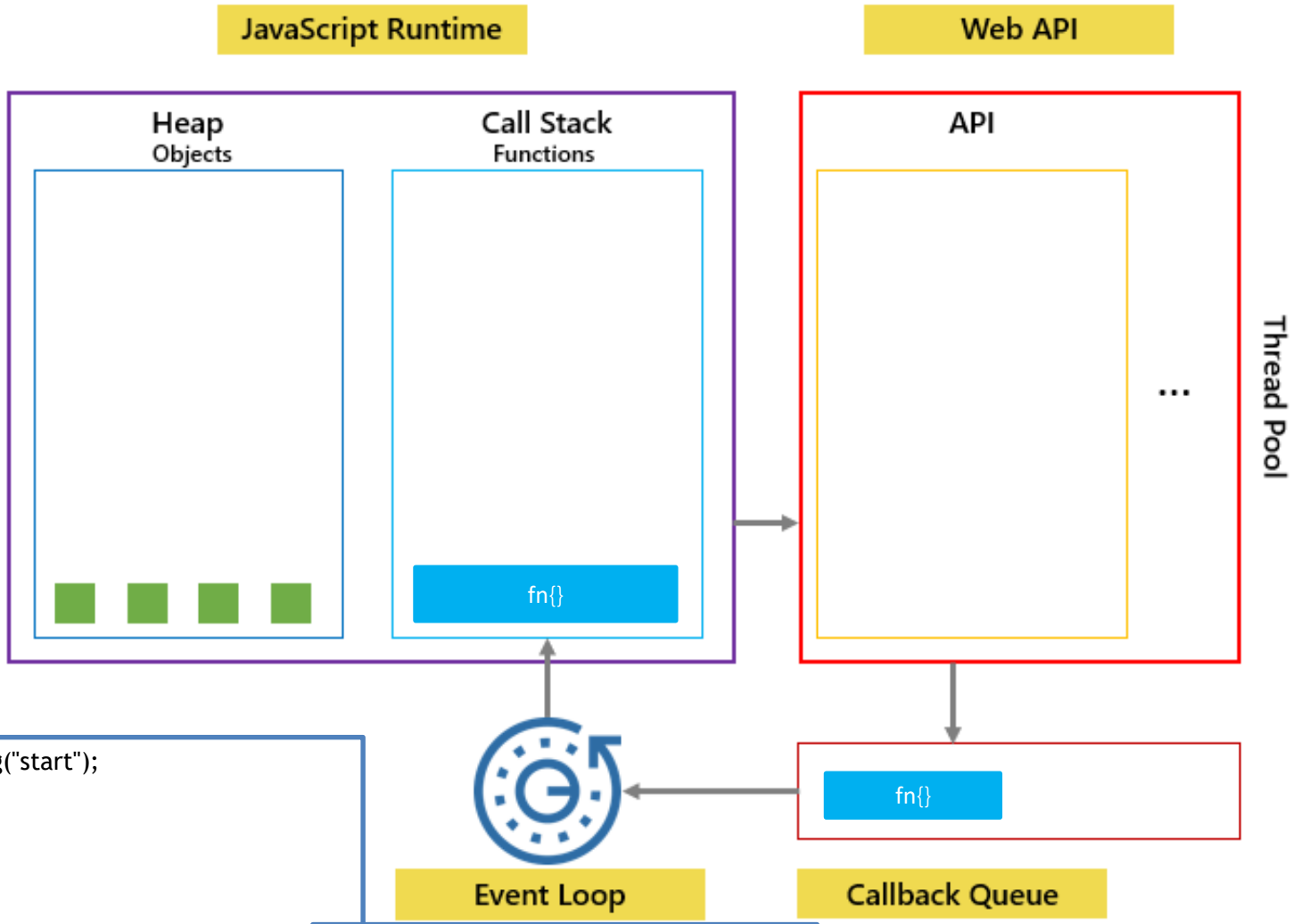

```
console.log("start");

wait5sec();

fun();

console.log("end");

setTimeout(function () {
    console.log("timeout after 1 sec");
}, 1000)
```

```
function fun() {
    setTimeout(function () {
        console.log("timeout immediately");
    }, 0);
}
```




```
console.log("start");

wait5sec();

fun();

console.log("end");

setTimeout(function () {
    console.log("timeout after 1 sec");
}, 1000)
```

```
function fun() {
    setTimeout(function () {
        console.log("timeout immediately");
    }, 0);
}
```




```
console.log("start");

wait5sec();

fun();

console.log("end");

setTimeout(function () {
    console.log("timeout after 1 sec");
}, 1000)
```

```
function fun() {
    setTimeout(function () {
        console.log("timeout immediately");
    }, 0);
}
```

JavaScript is the single-threaded programming language.

The JavaScript engine has only one call stack so that it only can do one thing at a time.

JavaScript execution is synchronous.
When executing a script,
the JavaScript engine executes code
from top to bottom,
line by line.

JavaScript also supports asynchronous operations through mechanisms like callbacks, events etc.

event loop to handle asynchronous operations

asynchronous nature is essential for handling non-blocking operations

Navigator

- The navigator object represents the browser application.
- This object allows scripts to get information about the browser like its type, version, language etc..
- Object Model reference:

  [window.]navigator
- All of its properties are read-only.

Navigator Properties & Methods

- Methods:
  - **javaEnabled()**
- Properties

https://developer.mozilla.org/en-US/docs/Web/API/Navigator

Name	Description	Syntax
appName (deprecated)	get the name of the browser	navigator.appName
appVersion (deprecated)	get the version of the browser	navigator.appVersion
language	get the default language of the browser	navigator.language
cookieEnabled	returns whether the browser allows cookies or not	navigator. cookieEnabled
platform (deprecated)	return the name of the OS	navigator.platform
vendor (deprecated)	Returns the vendor name of the current browser	navigator.vendor

HTML5 New API →

geolocation

returns a Geolocation object allowing accessing the location of the device

Example!

Location

- The Location object is part of a Window object.
- The location Object refers to the current URL.
- Location contains information about the current URL of the browser. The most common usage of Location is simply to use it to automatically navigate (redirect) the user to another web page.
- It has a set of properties to hold the different components of the URL
- Object Model Reference:

  [window.]location

```
<script type="text/javascript">
    window.location=http://www.google.com
< /script>
```

Location Properties

Name	Description	Syntax
href	is the default property of the location object, returns the entire URL	location.href
protocol	represents the protocol of the URL.	location.protocol
hostname	specifies the host name	location.hostname
port	specifies the communication port.	location.port
host	is a combination of the host name and port	location.host
pathname	is the directory to find the document on the host, and the name of the file	location.pathname
search	specifies the queryString	location.search

Location Methods

- *replace* method loads the specified URL over the current history entry.

  location.replace(URL)

- *reload* method Reloads the current document over the current history entry.

  location.reload()

- *assign* method is almost the same as *replace* method.
  The difference is that it creates an entry in the browser's history list, while replace() doesn't.

  location.assign(URL)

- *toString* method returns a string representation containing the whole URL

  location.*toString* ()

Example!

History

- The history Object lets you send the user to somewhere in the history list from within a JavaScript program.
- Object Model reference:

  [window.]history

- Properties:

length

- Methods:

back()	**forward()**	**go()**

Example!

Assignments