Client-side Technologies

Eng. Niween Nasr El-Den
SD & Gaming CoE
iTi

Day 5

Basics of JavaScript

Browser Object Model

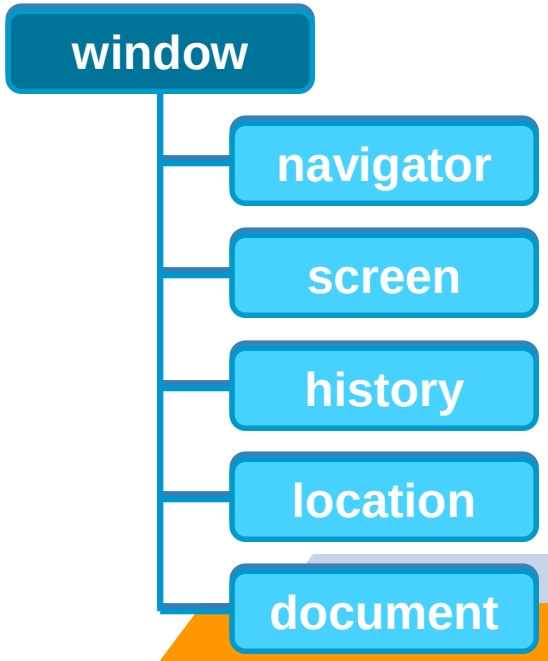
BOM

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.



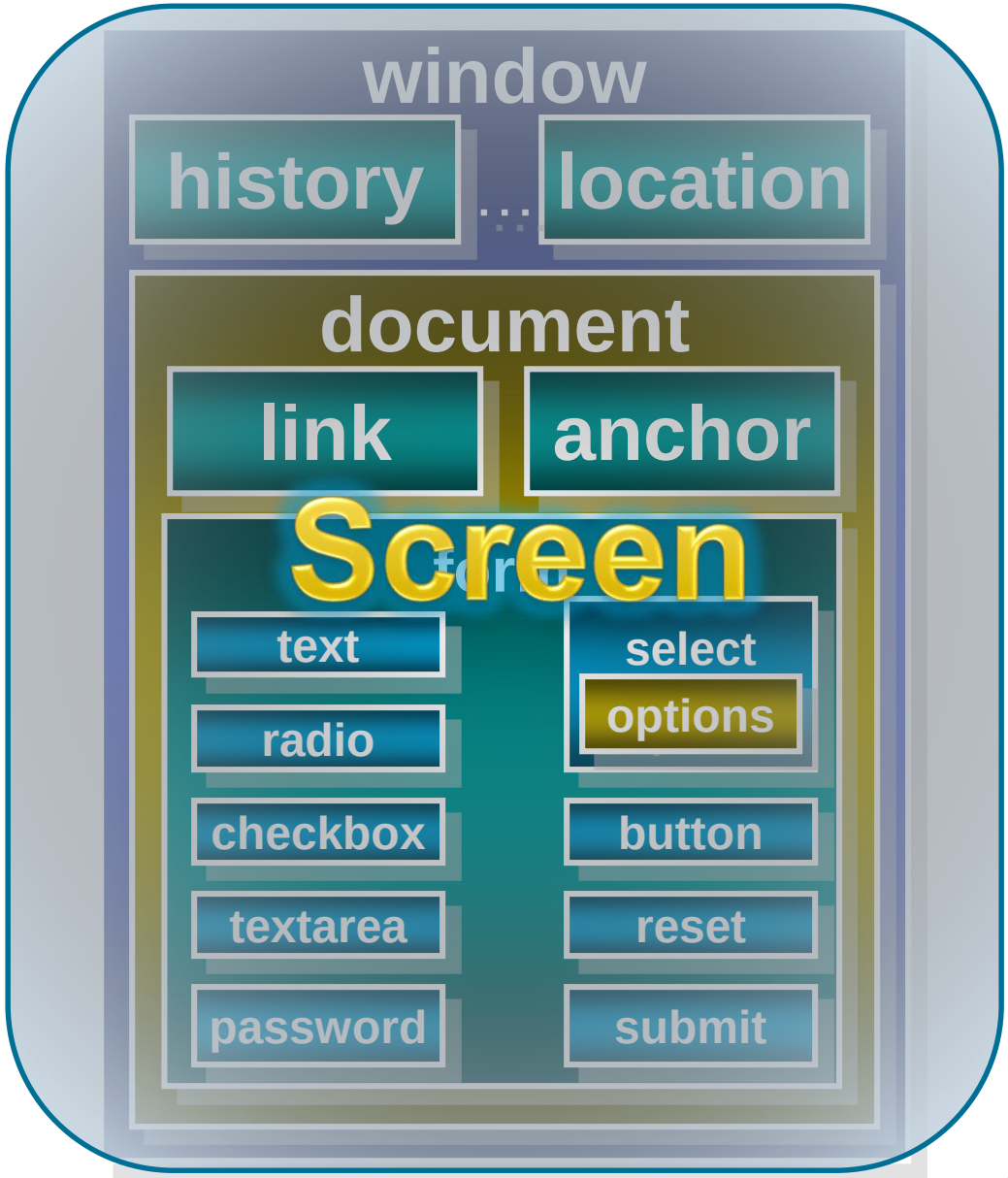
BOM

- Using the **BOM**, developers can move the window, change text in the status bar, 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.

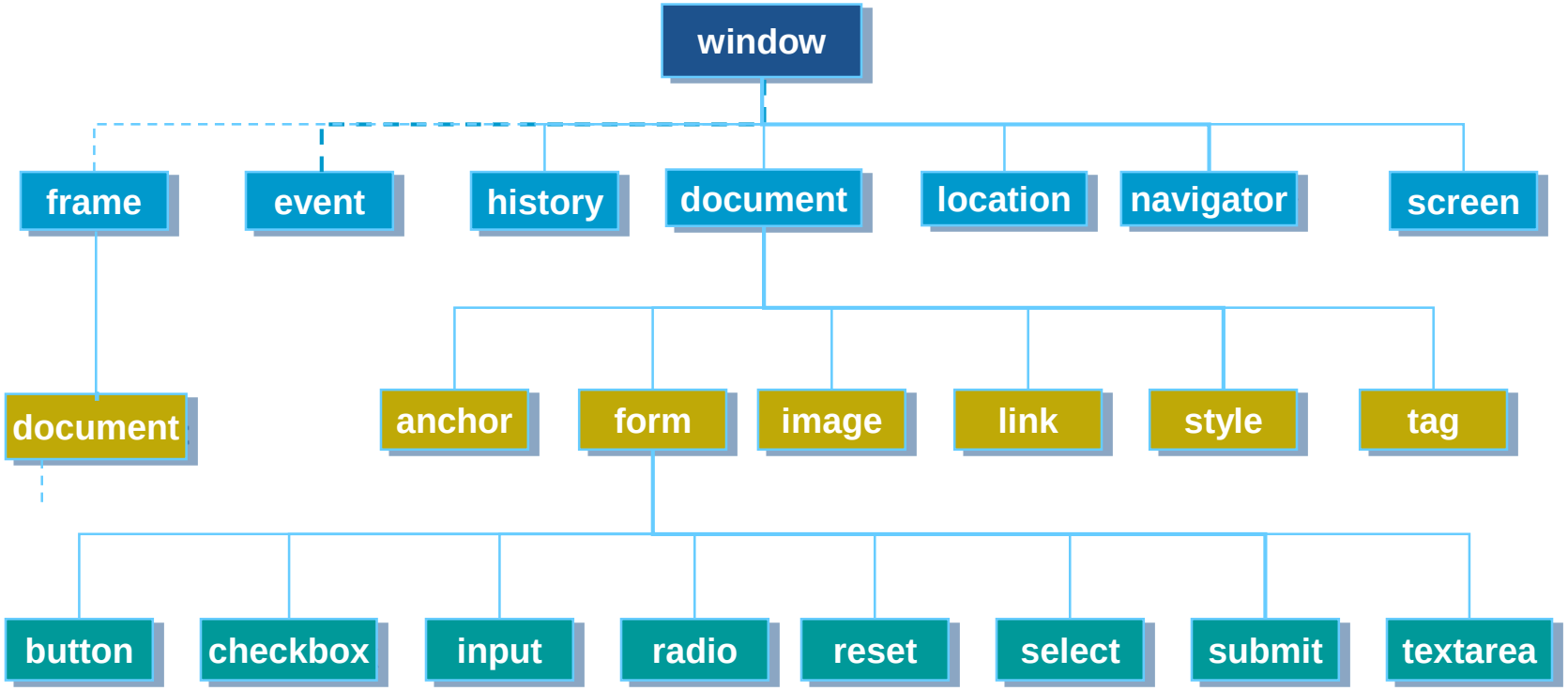
JavaScript Top Object Model Hierarchy

- 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
opener	Returns a reference to the window object that created this current window Note: If the current window has no opener, this method returns NULL.	**window.opener**
closed	**Specifies whether or not a window has been closed. Returns true if the window is closed**	**window.closed**
innerHeight	Gets the height of the content area of the browser window including, if rendered, the horizontal scrollbar.	window.innerHeight
innerWidth	Gets the width of the content area of the browser window including, if rendered, the vertical scrollbar.	window.innerWidth
outerheight / outerwidth	These properties determine the dimensions, in pixels, of the outside boundary of browser window.	**window.outerheight** **window.outerwidth**

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_setInterva)
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!

Browser Engine & JavaScript

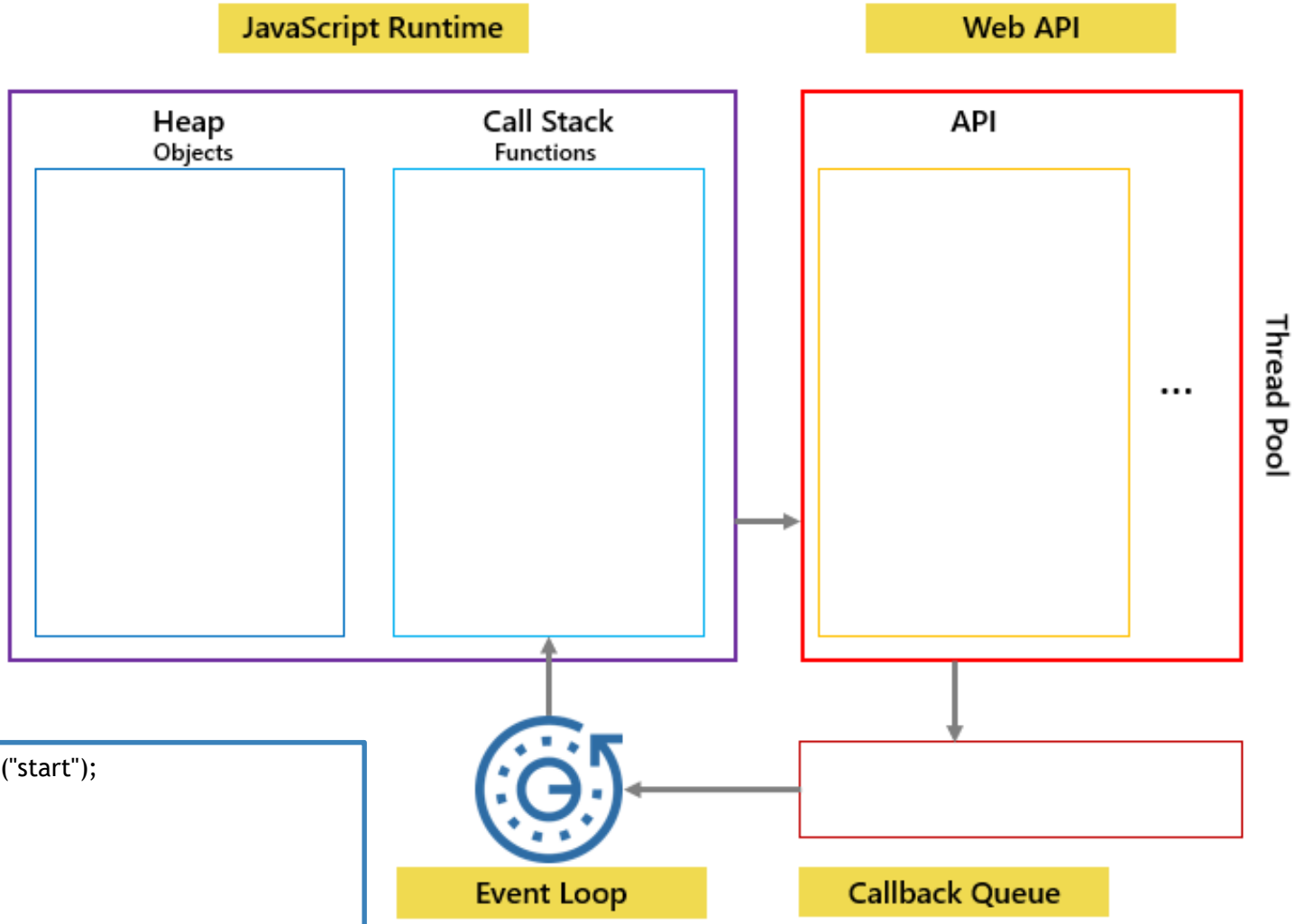
https://en.wikipedia.org

- **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.

Example

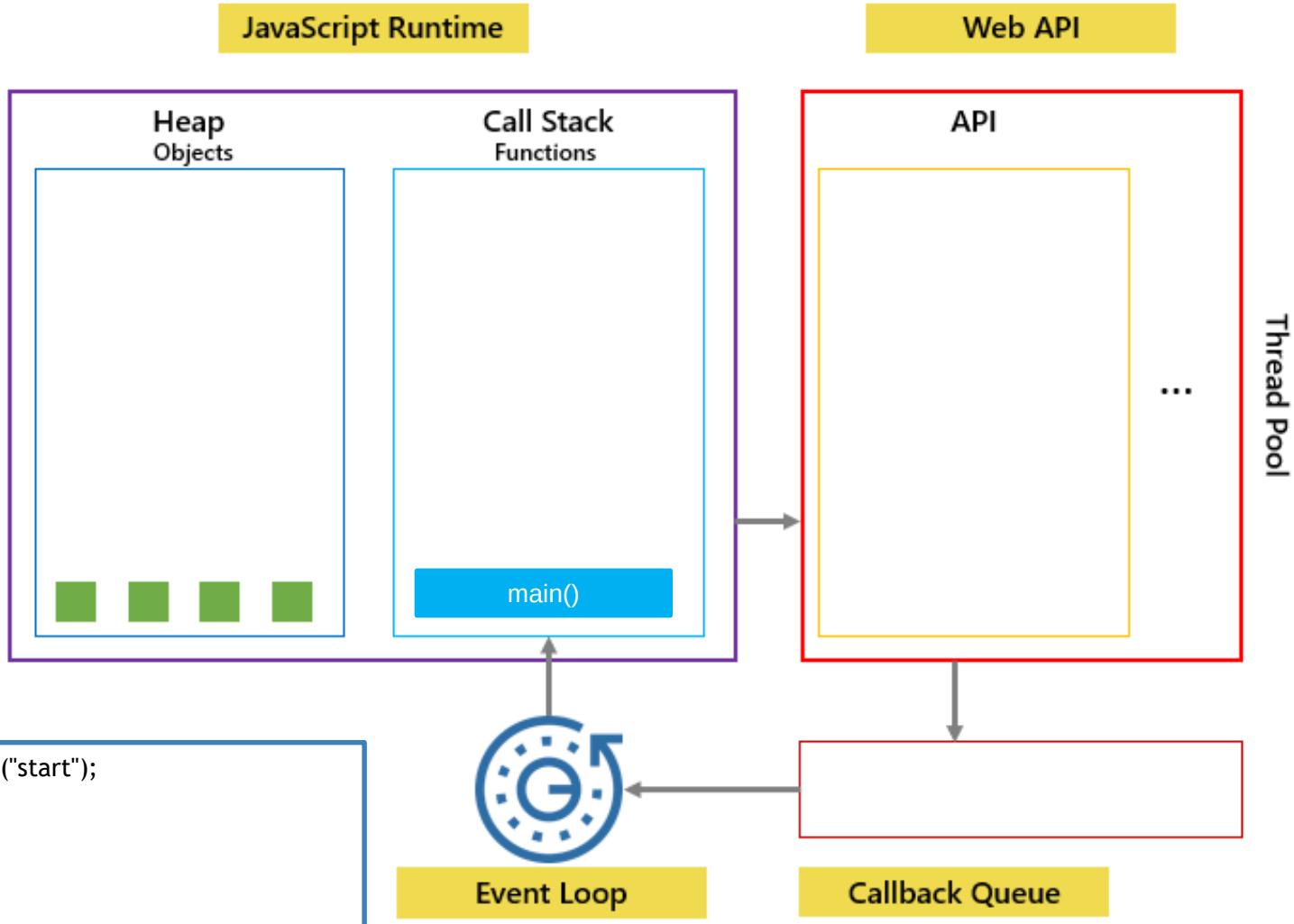
```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



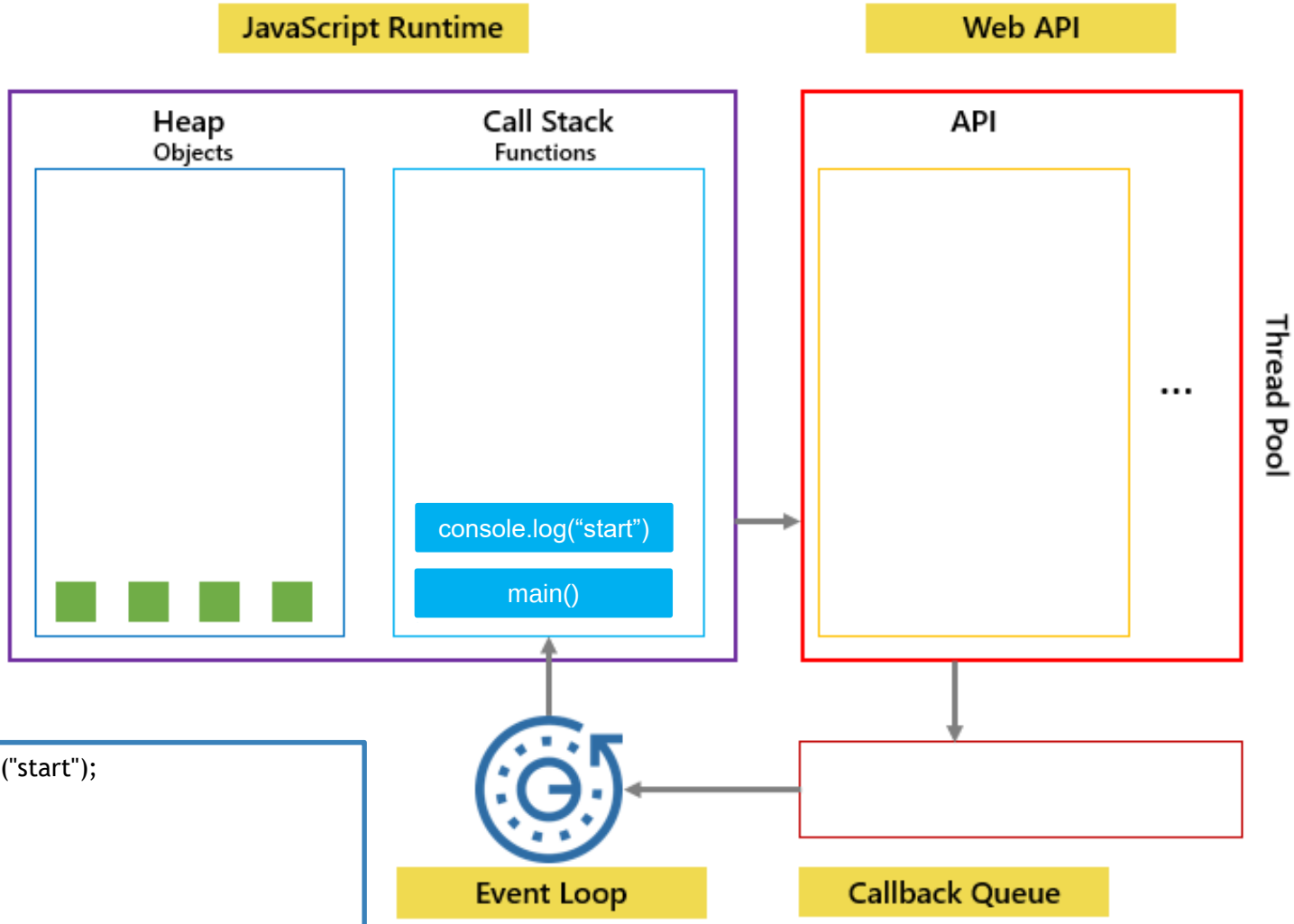

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



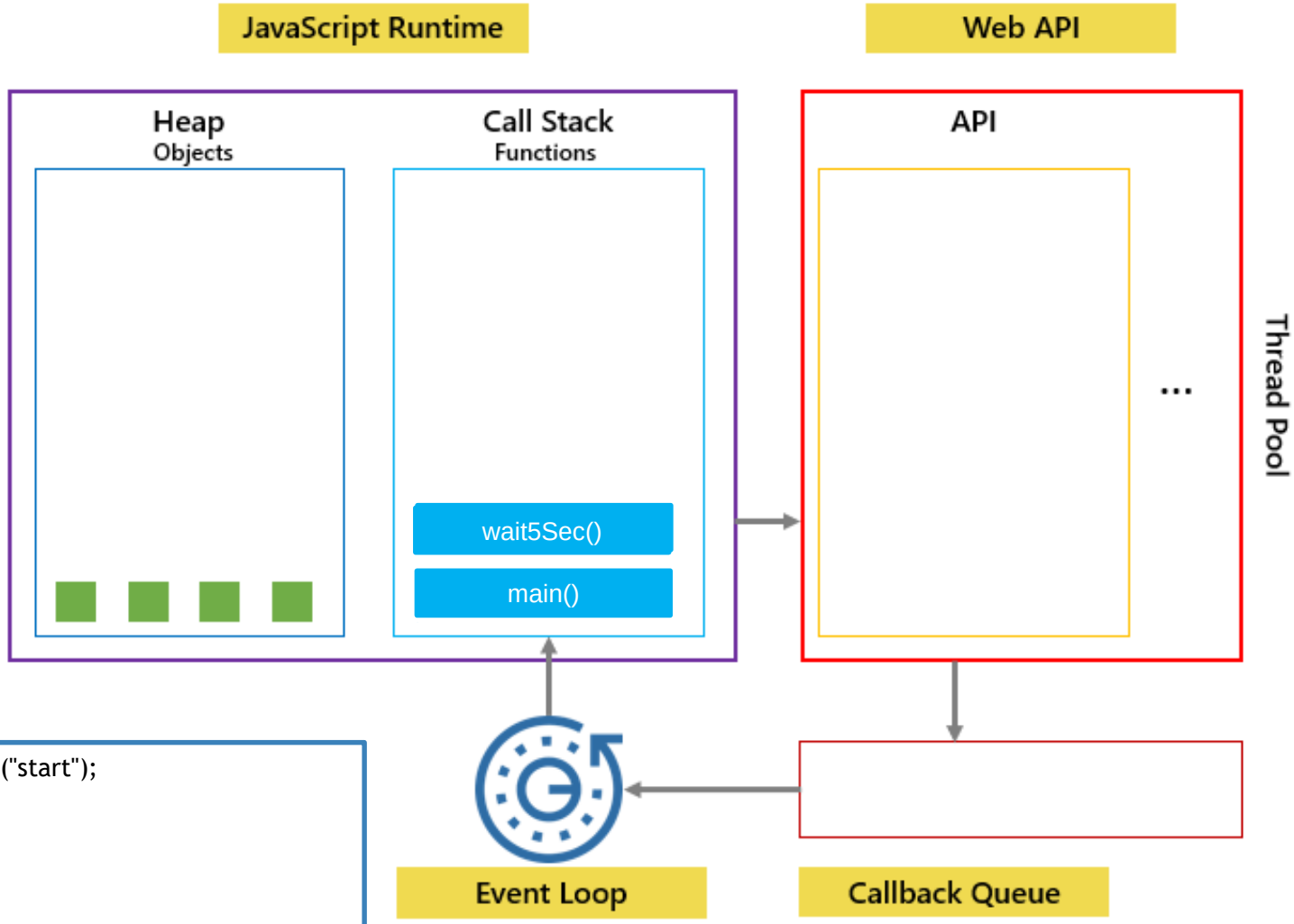

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



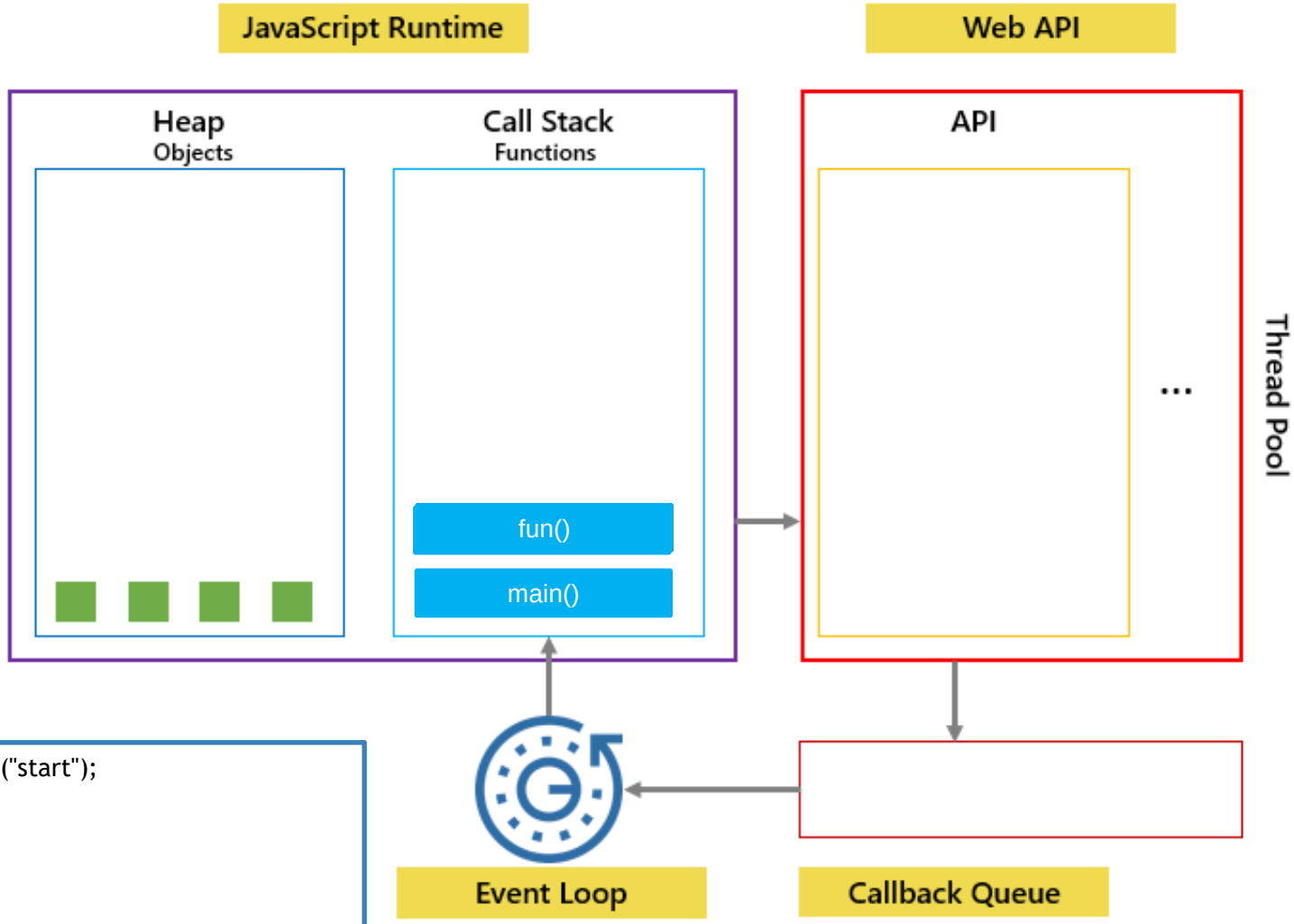

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



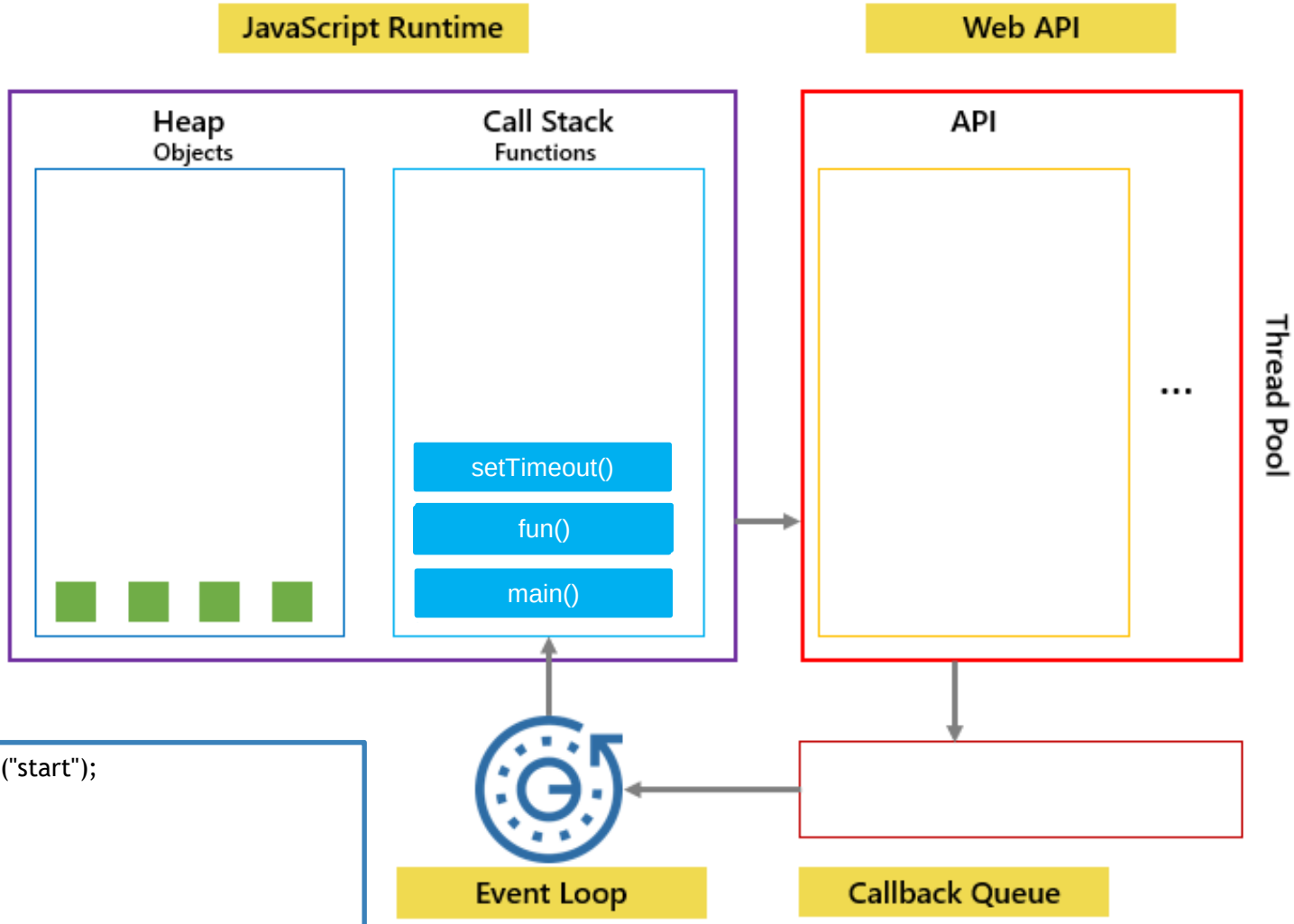

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



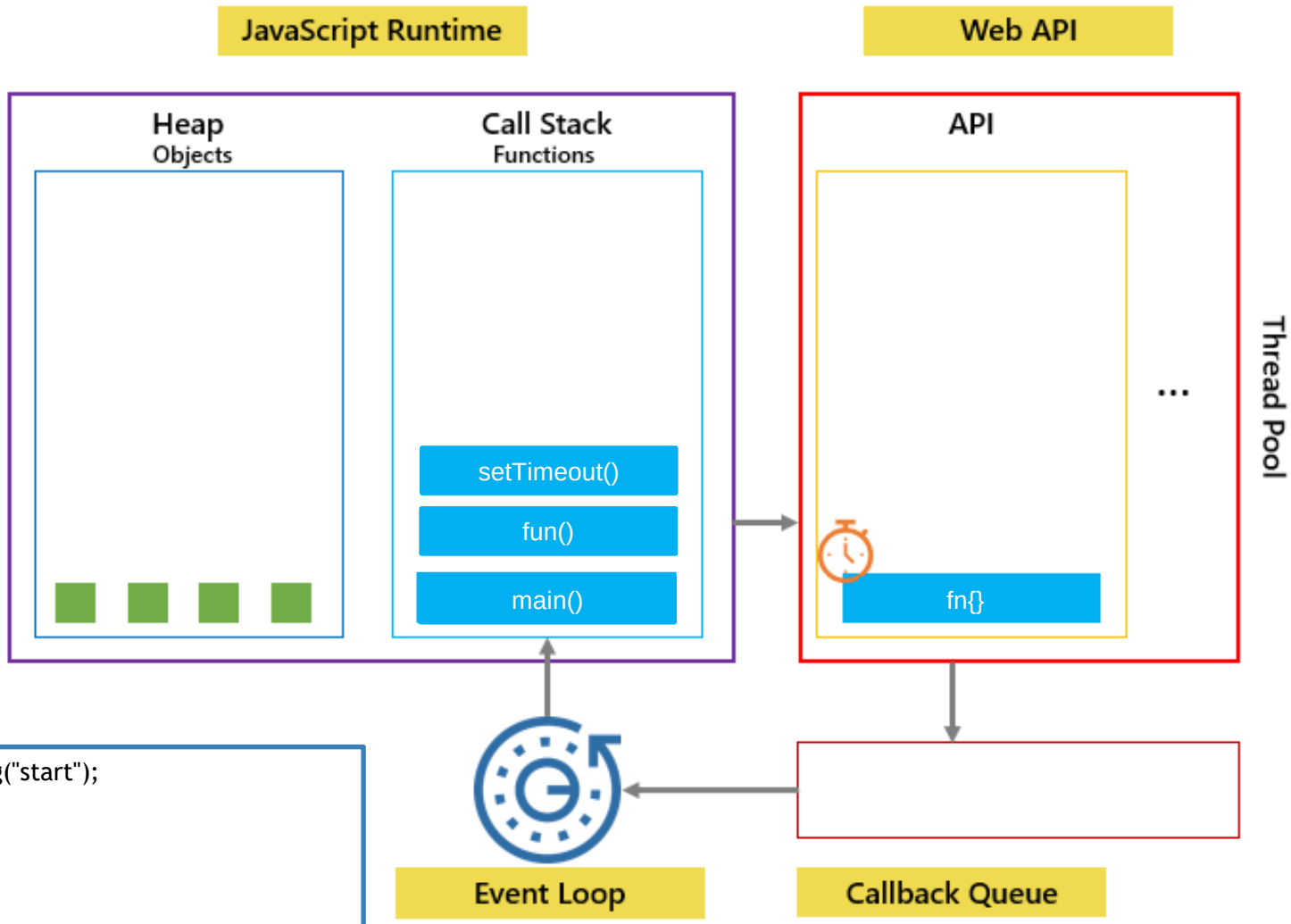

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



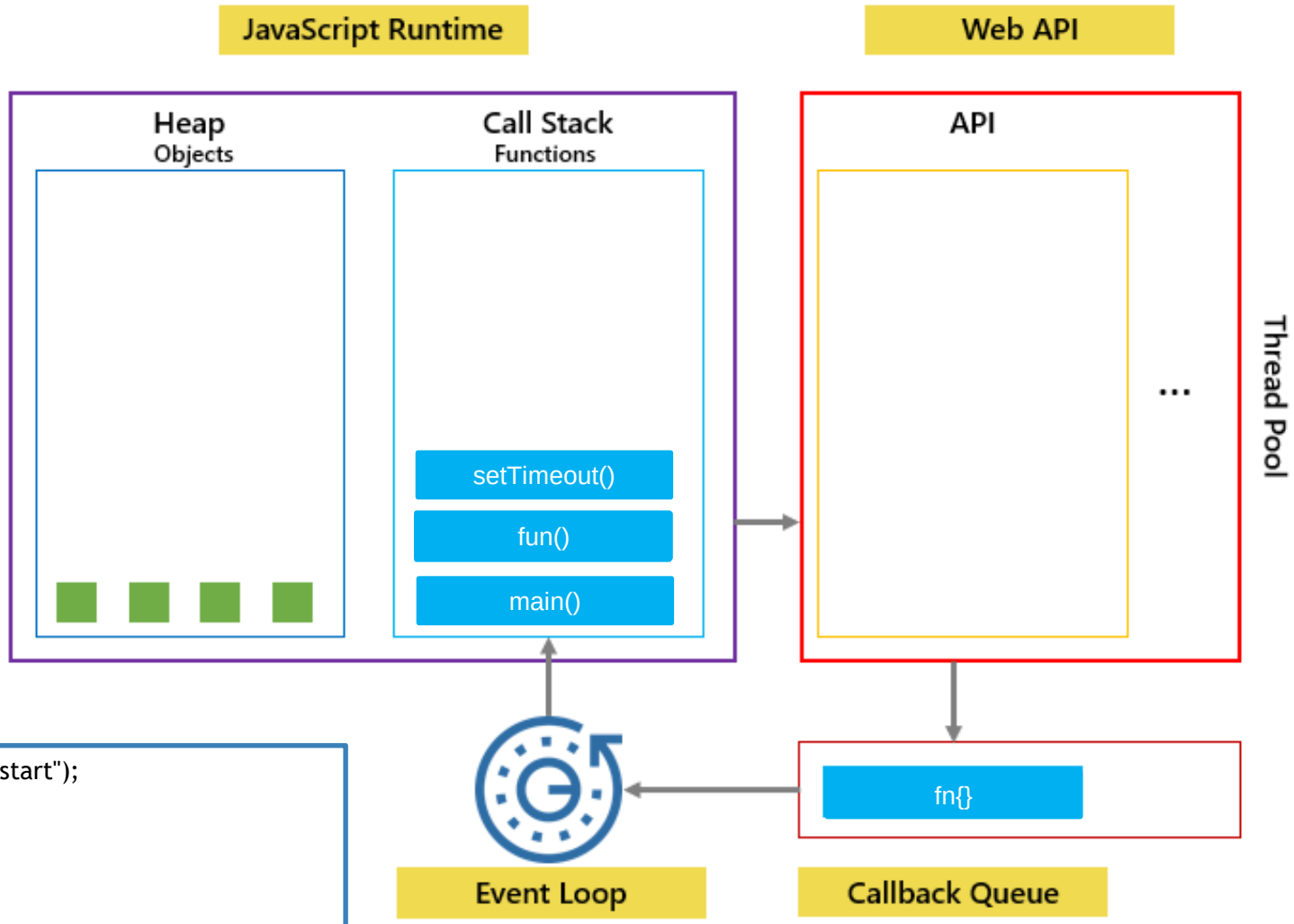

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



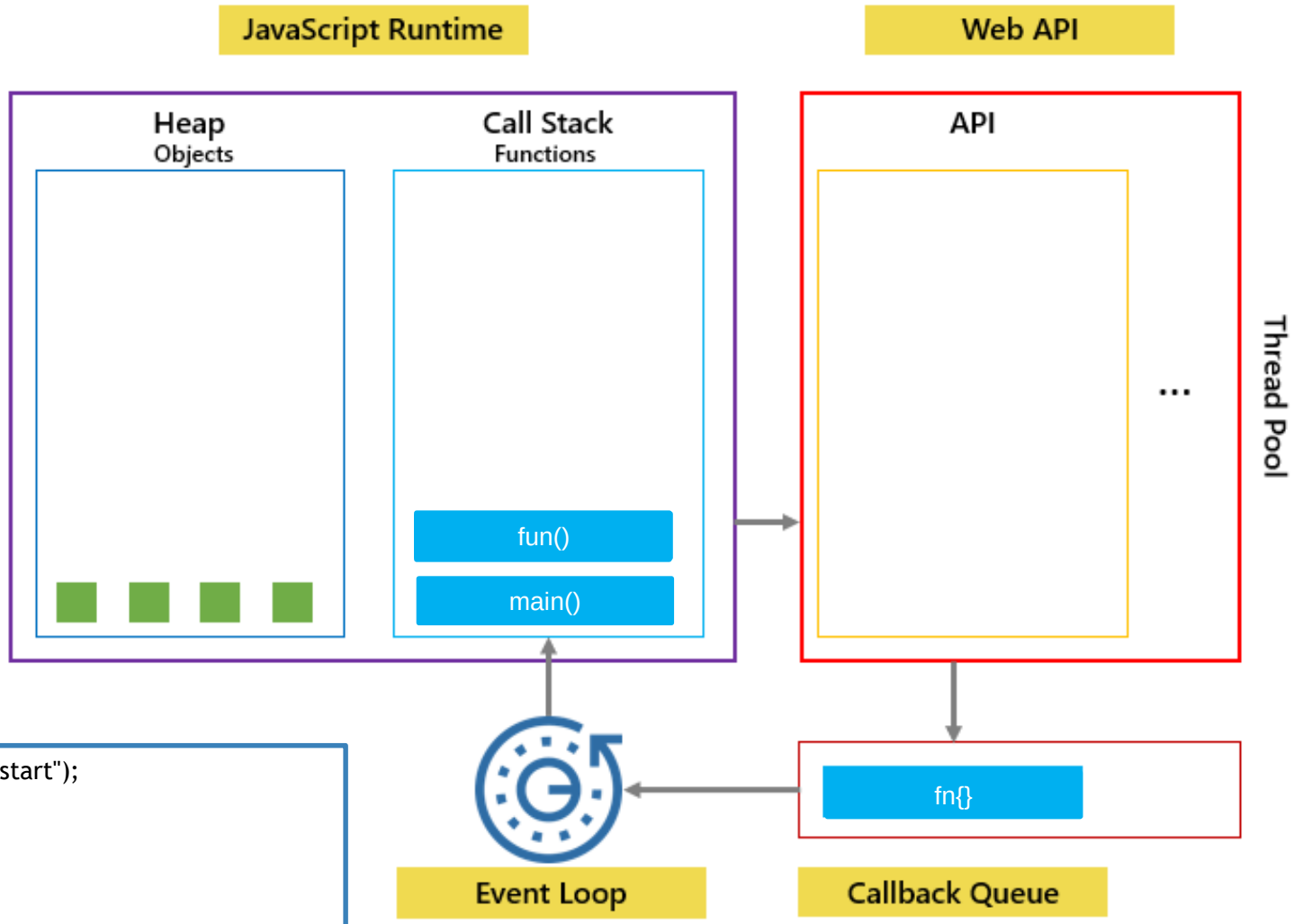

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



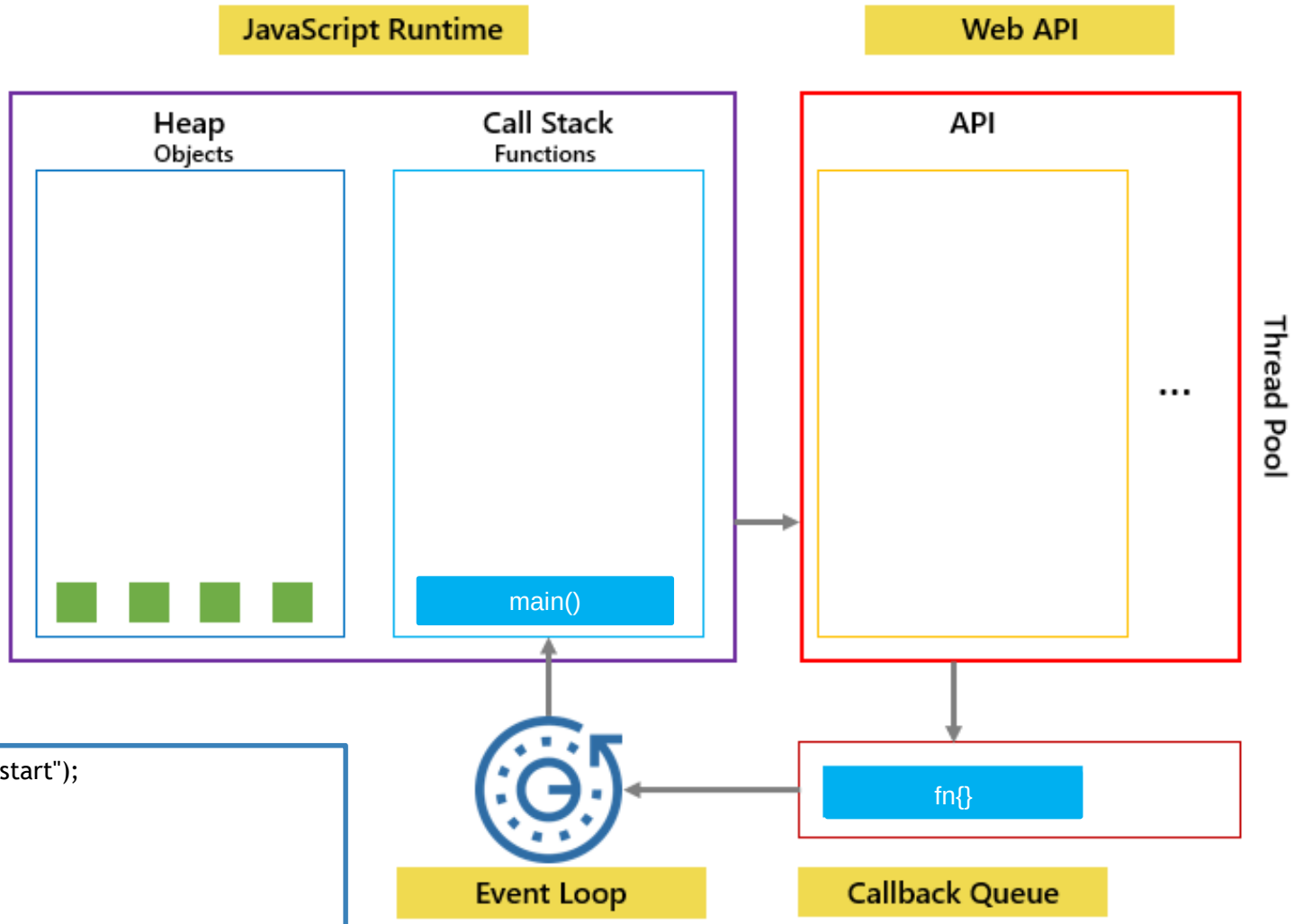

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



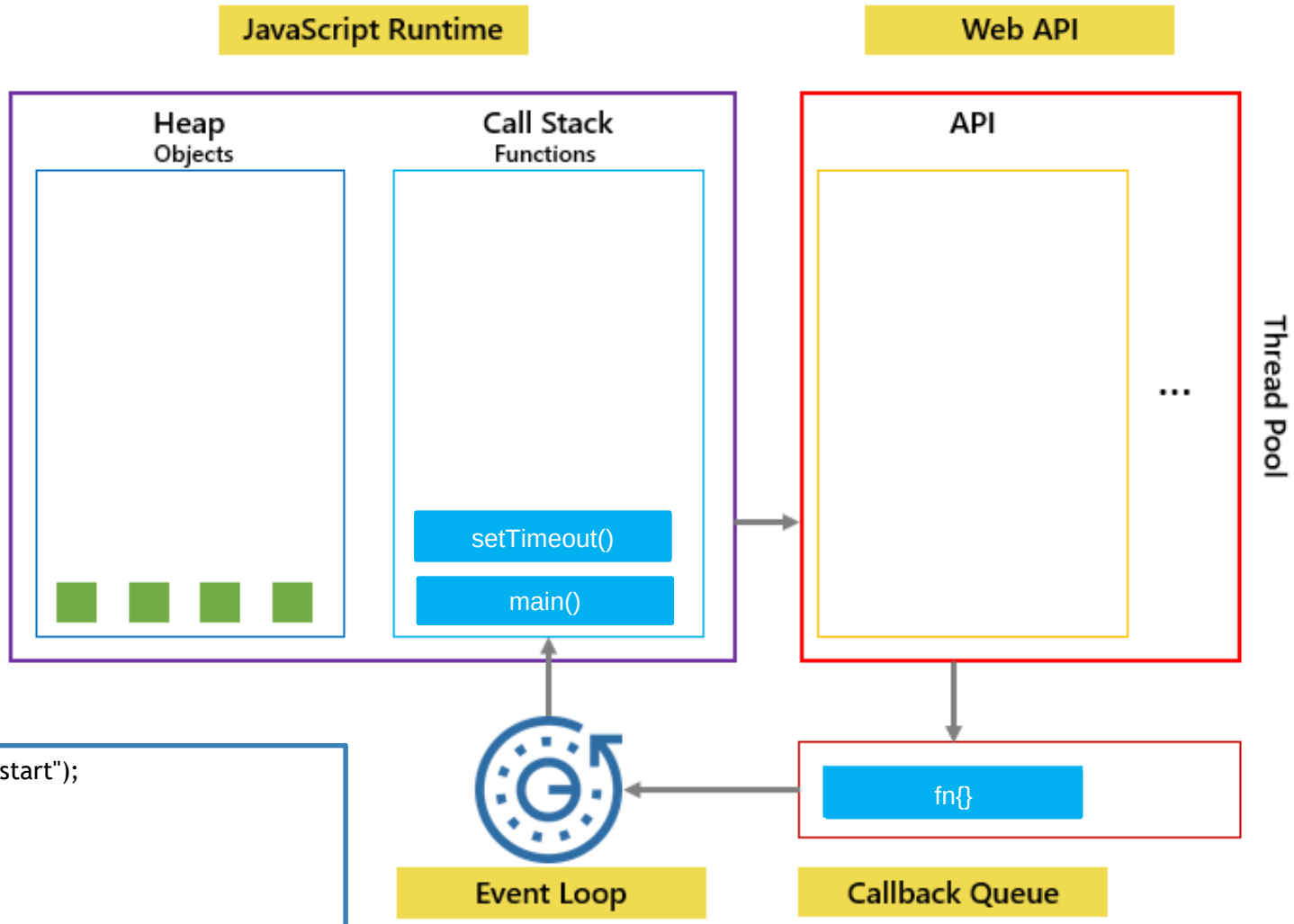

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



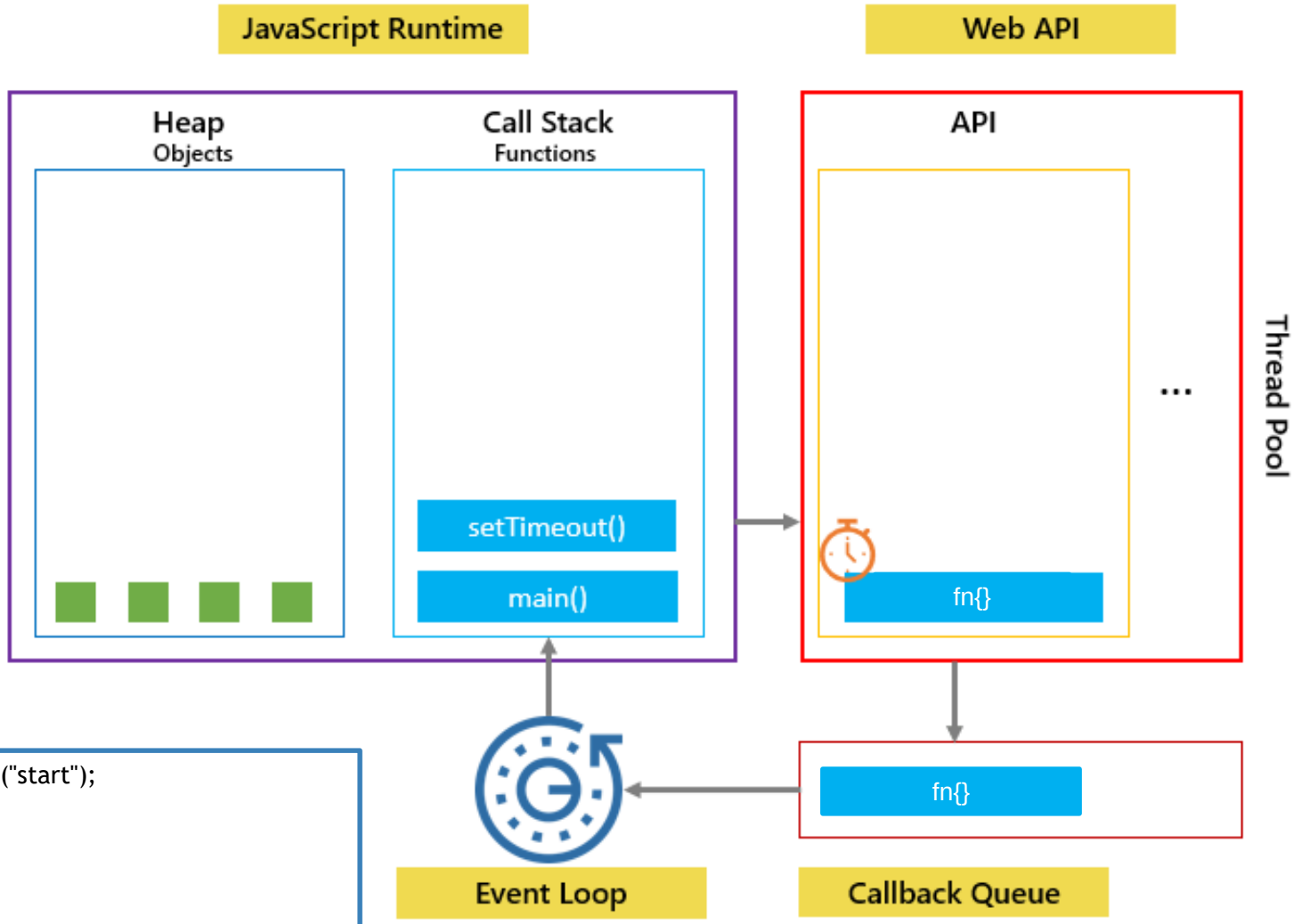

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



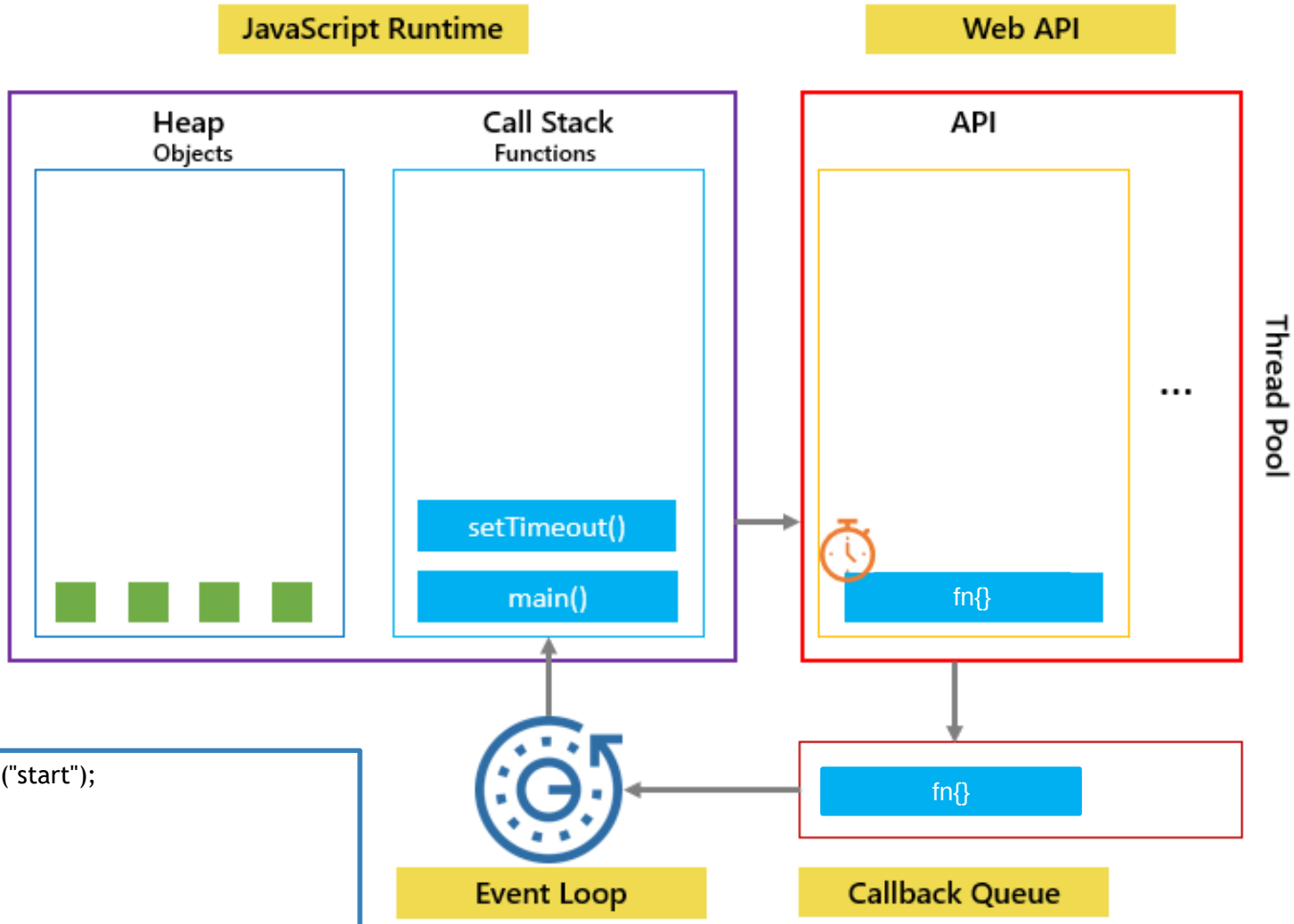

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



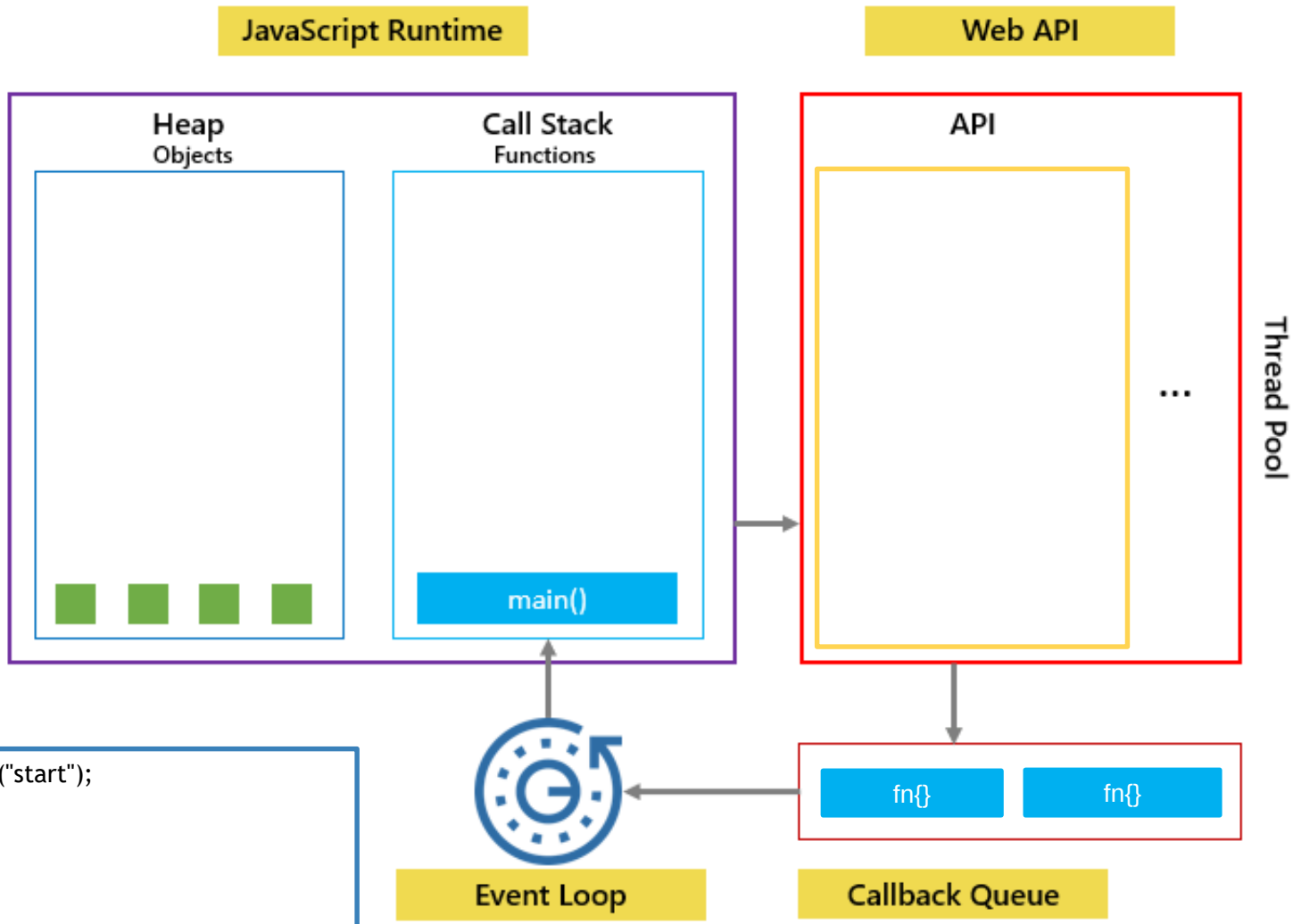

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



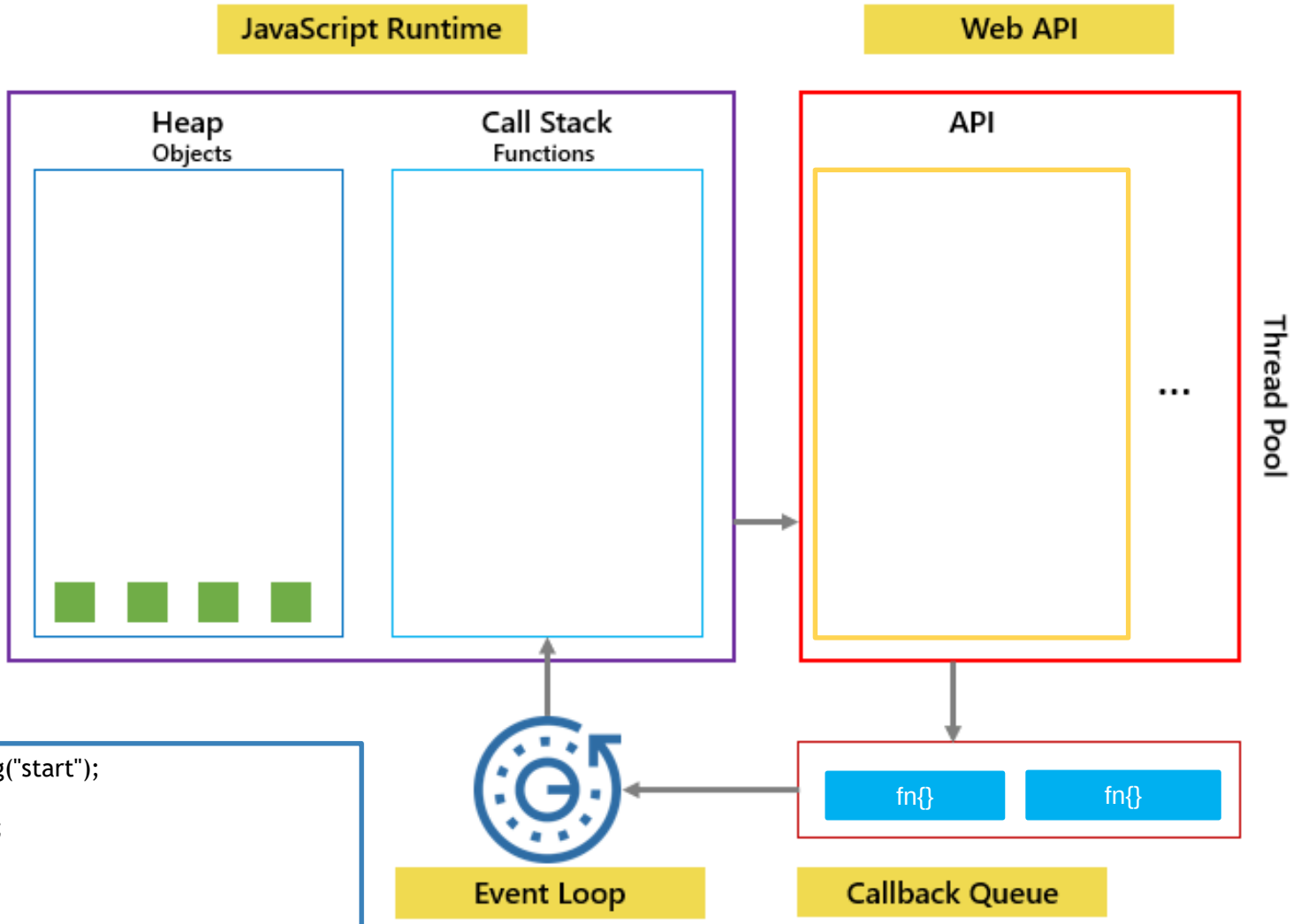

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



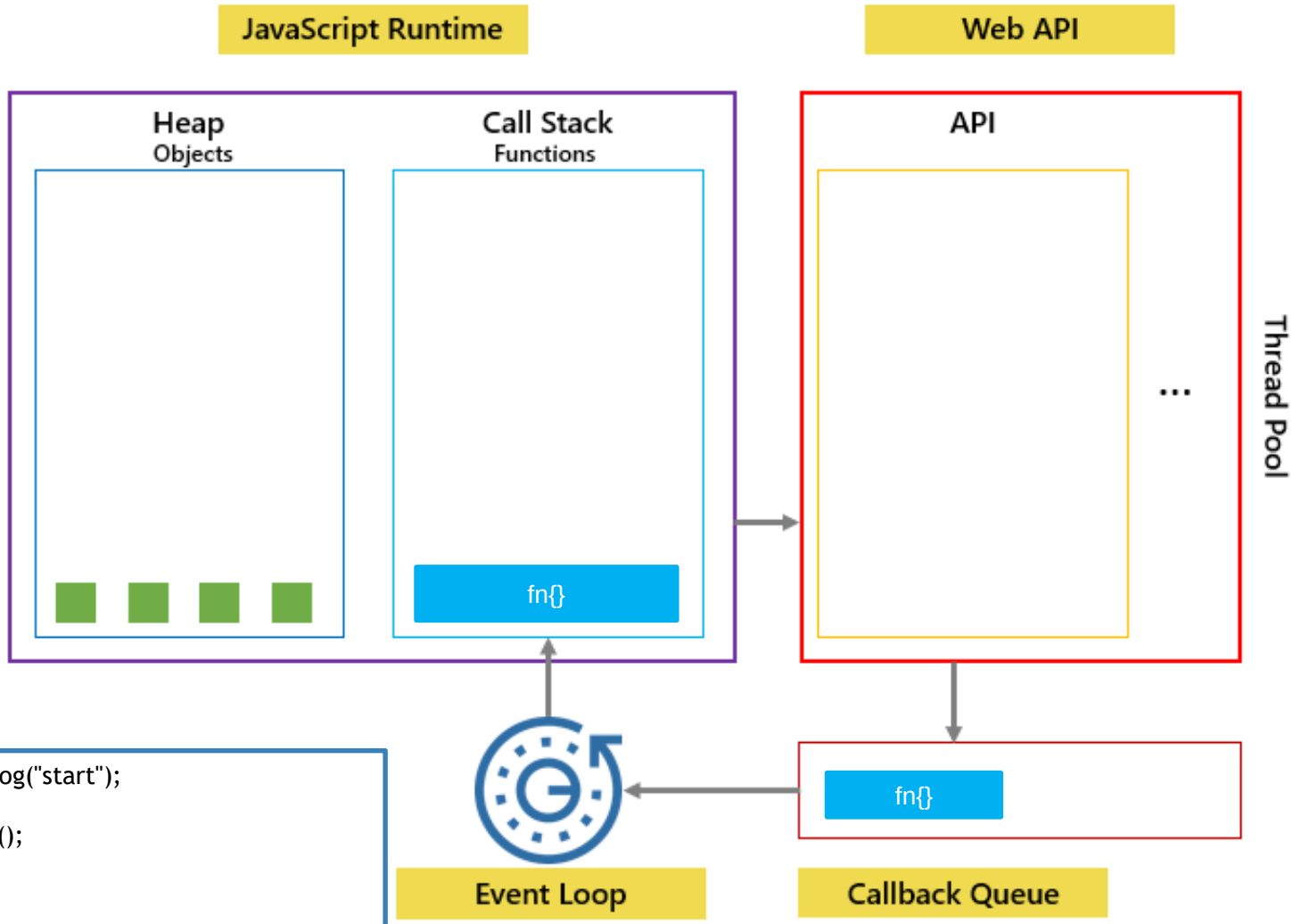

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



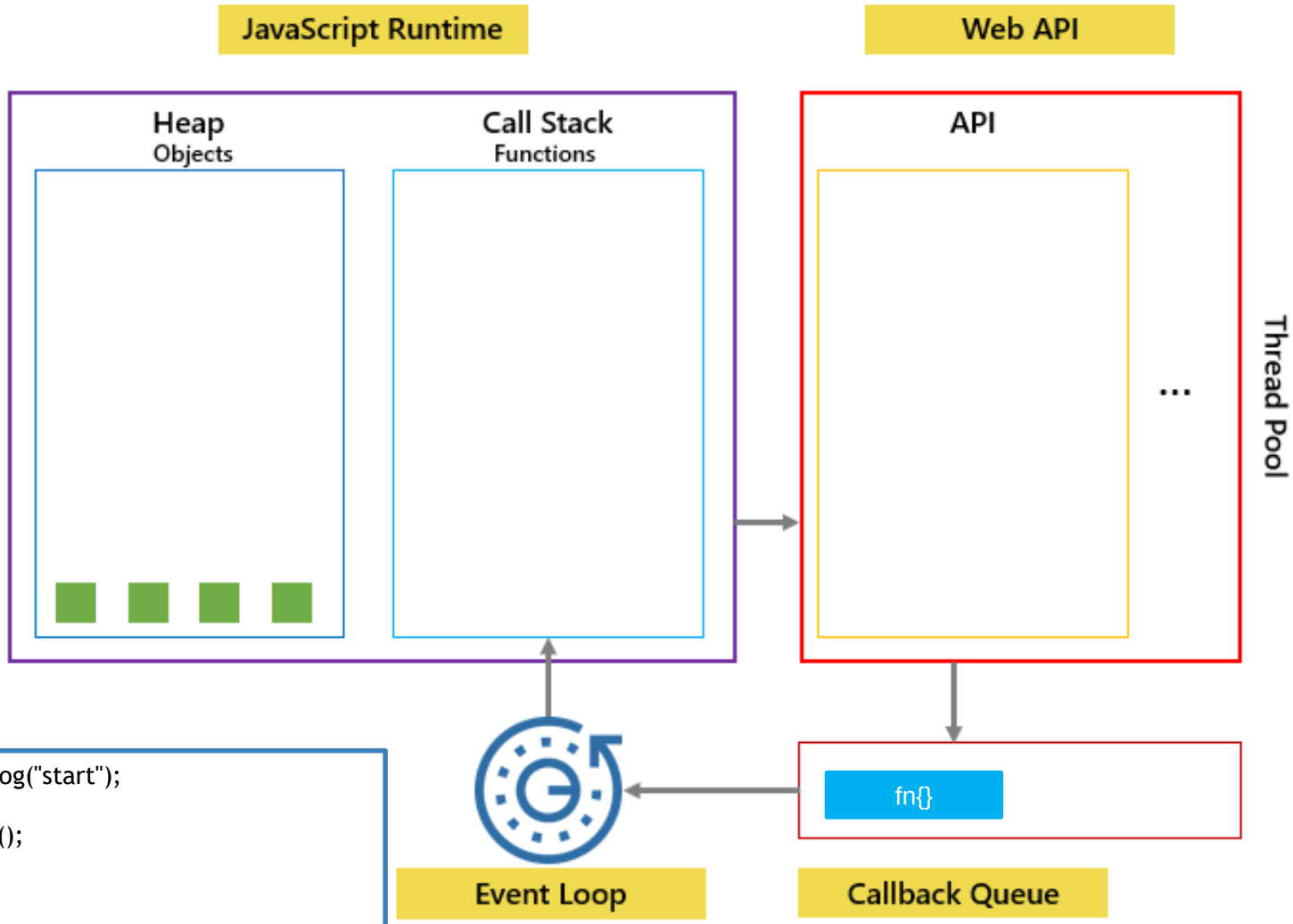

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



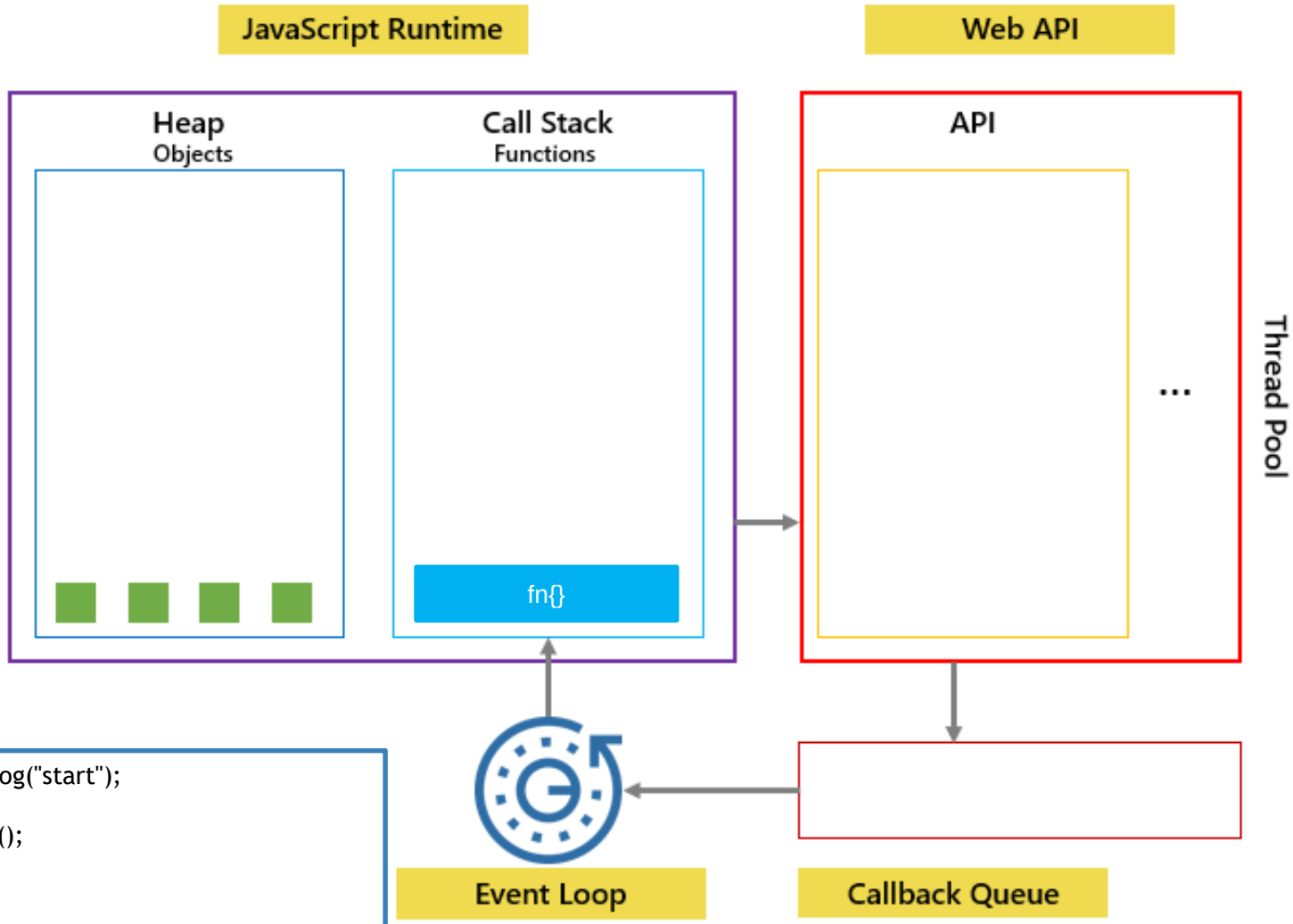

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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



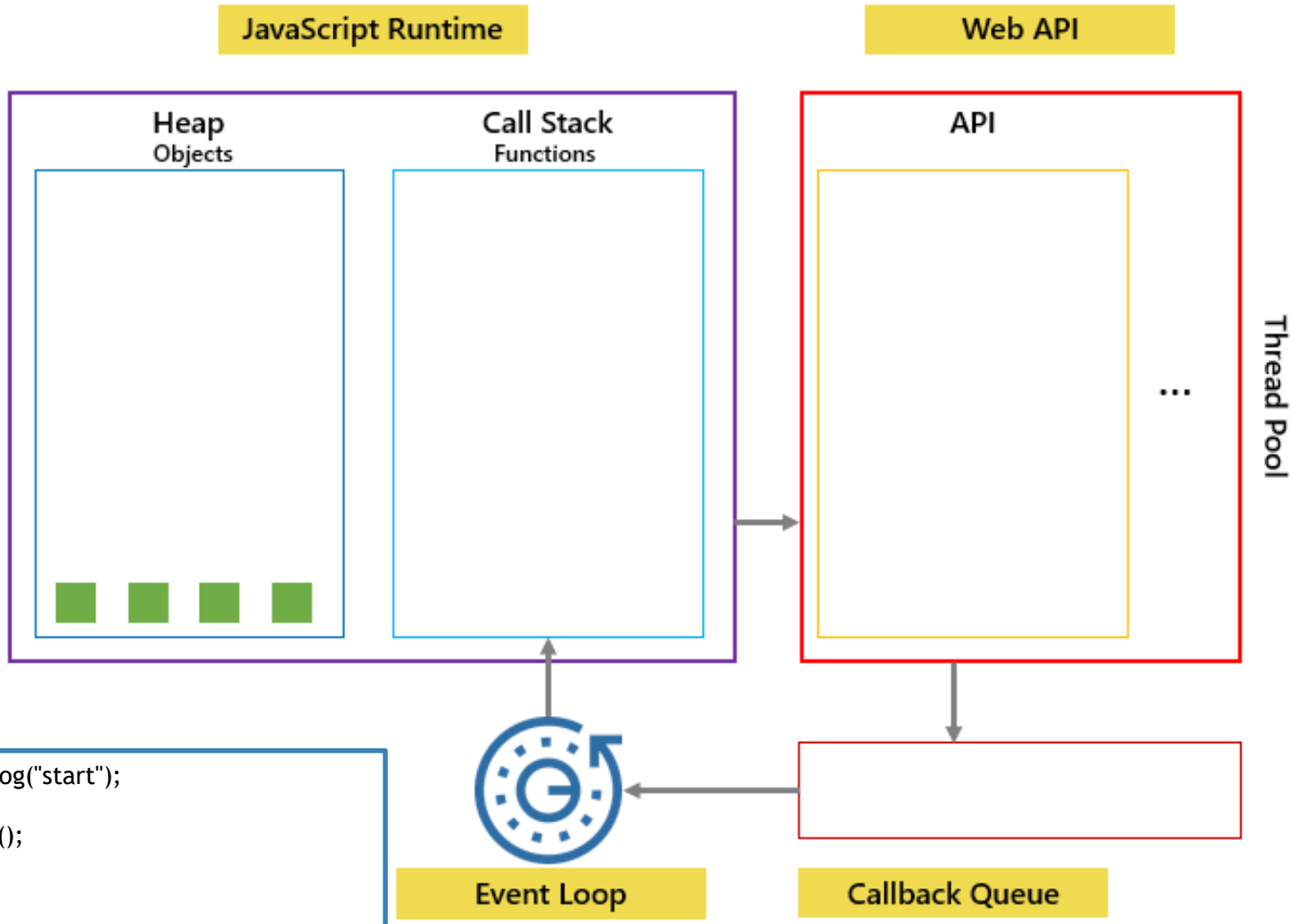

```
console.log("start");

wait5sec();

fun();

console.log("end");

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

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




```
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 (concurrently).

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

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

Properties

Name	Description	Syntax
appName (deprecated)	get the name of the browser	navigator.appName
appVersion (deprecated)	get the version of the browser	navigator.appVersion
language browserLanguage IE	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

Methods

javaEnabled()

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=“documentURL” location.href=“documentURL ”
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!

Document Object Model

DOM

DOM

- DOM Stands for Document Object Model.
- W3C standard.

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

- Its an API that interact with documents like HTML, XML.. etc.
- Defines a standard way to access and manipulate HTML documents.
- Platform independent.
- Language independent

DOM

- The **document** object in the **BOM** is the top level of the **DOM** hierarchy.
- DOM is a representation of the whole document as nodes and attributes.
- You can access each of these nodes and attributes and change or remove them.
- You can also create new ones or add attributes to existing ones.

DOM is a subset of BOM.

In other word: **the document is yours!**

DOM

methods and properties, allows accessing any element on the page, modify, delete or remove elements, or add new ones.

Document Object

- The *document* object represents the entire HTML document and can be used to access all elements in a page.
- A *page* is what appears within the browser window.
- So, every window is associated with a document object.
- Document Object has its own set of Properties, Collections, Methods & Event handlers.

The document object in the BOM is the top level of the DOM hierarchy.

Document Methods

Method	Description
write()	Writes one or more HTML expressions to a document in the specified window.
writeln()	Writes one or more HTML expressions to a document in the specified window and follows them with a new line character.

Example!

Document Methods

for accessing document elements

Method	Description
getElementById(“id")	For accessing any element on the page via its ID attribute
getElementsByName(“name”)	Returns a collection of objects with the specified name
getElementsByTagName(“tagName”)	Returns a collection of objects with the specified tagname
getElementsByClassName(“className”)	Returns a collection of objects with the specified classname

Example!

New HTML5 Selectors

- In HTML5 we can select elements by ClassName

```
var elements = document.getElementsByClassName('entry');
```

- Moreover there's now possibility to fetch elements that match provided CSS syntax

```
var elements =  document.querySelectorAll(".someClasses)");
```

```
var elements = document.querySelectorAll("div,p");
```

```
var elements = document.querySelector("#someID");
```

```
var first_td =   document.querySelector("span");
```

DOM Accessing method returns a JavaScript object that represents the HTML markup element.
This Object’s properties represent the markup’s attribute

We can access any markup content via innerHTML, innerText, or textContent properties

Document Properties

Property	Description
bgColor	A string that specifies the background color.
fgColor	A string that specifies the text color.
linkColor	The color of text for a link that the user has not yet visited.
vlinkColor	The color of text for a link that the user has already visited.
alinkColor	The color of text for a link that the user clicks.
title	A string that specifies the contents of the TITLE tag.
cookie	A string containing the name/value pair of cookies in the document.

Document Collection

- links, anchors, images, forms are collection/array containing all occurrences of those objects within the document.
- Since they are treated as arrays, they have the *length* property which specifies the number of entries in the collection/array.
- To access a specific entry in any of these collections, we can use either their index or name.

Document Collection

Collection	Description
forms[ ]	An array containing an entry for each form in the document
images[ ]	An array containing an entry for each image in the document
anchors[ ]	An array containing an entry for each anchor in the document.
links[ ]	An array containing an entry for each link in the document.

Document Collection

- An item from an object collection can be referenced in one of the following ways:

  1. collection[i]
  2. collection.item(i)
  3. collection.namedItem(id)
  4. collection[“name”]
  5. collection[“id”]
  6. collection.name

- Example: document.forms[0]

  document.forms[“myForm”]

  document.forms[“frmid”]

  document.myForm

Document Event Handler

onclick
ondblclick
onkeydown
onkeypress
onkeyup
onmousedown
onmouseup

Image

- The Image object is an image on an HTML form, created by using the HTML 'IMG' tag.
- Any images created in a document are then stored in an array in the document.images property.
- Image Object has its own set of Properties, Methods & Event handlers.

Image

- **Object Model Reference:**

[window.]document.imageName
[window.]document.imageID
[window.]document.images[i]
[window.]document.images[imageName]

```
<html>
<body>
 …..
<img name="img1" src="img1.jpg">
<img name="img2" src="img2.jpg">
…
</body>
</html>
```

```
document.img1.src="img1.jpg"
document.images[0].src="img1.jpg"
```

```
document.img2.src="img2.jpg"
document.images[1].src="img2.jpg"
```

Image Properties & Event handlers

Properties

name	id	src	height	width

Event handlers

onabort	onload	onerror	onclick	ondblclick	onmouseover

Example!

Assignment