KINGSLAND
UNIVERSITY

DOM Manipulations

Create / Delete DOM Elements, Handle Browser Events

Table of Contents

Event Loop

JS Approach to I/O

- Single threaded language
- HTTP requests
- DB
- Memory and disk read/write

Don't Make the Thread Wait

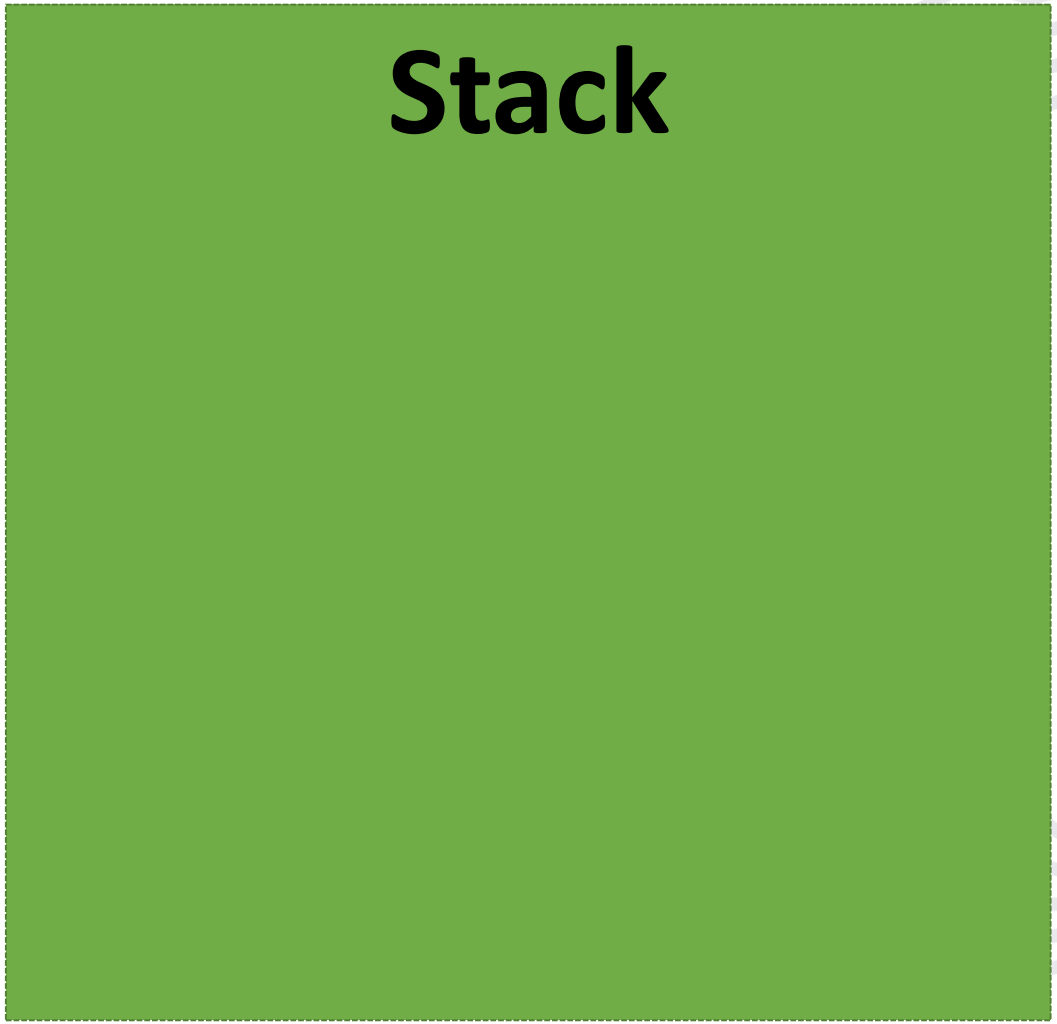
- Blocking thread requests
- Register a callback
- Handle multiple concurrent operations on one thread

How It's Handled in JS?

- Message queue
- Event loop

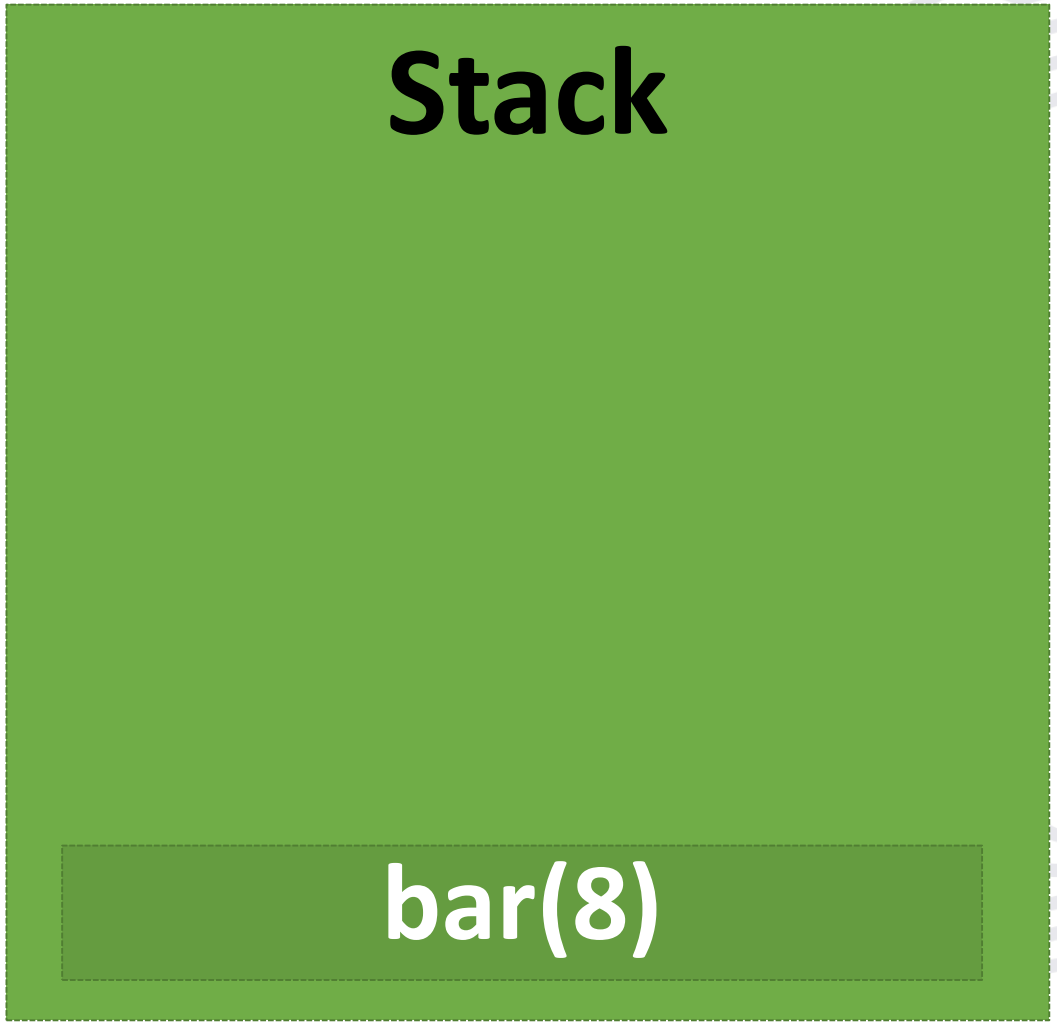
Stack calls

```
function foo(x) {
    return x * x;
}
function bar(y) {
    return foo(y + 2);
}

bar(8);
```



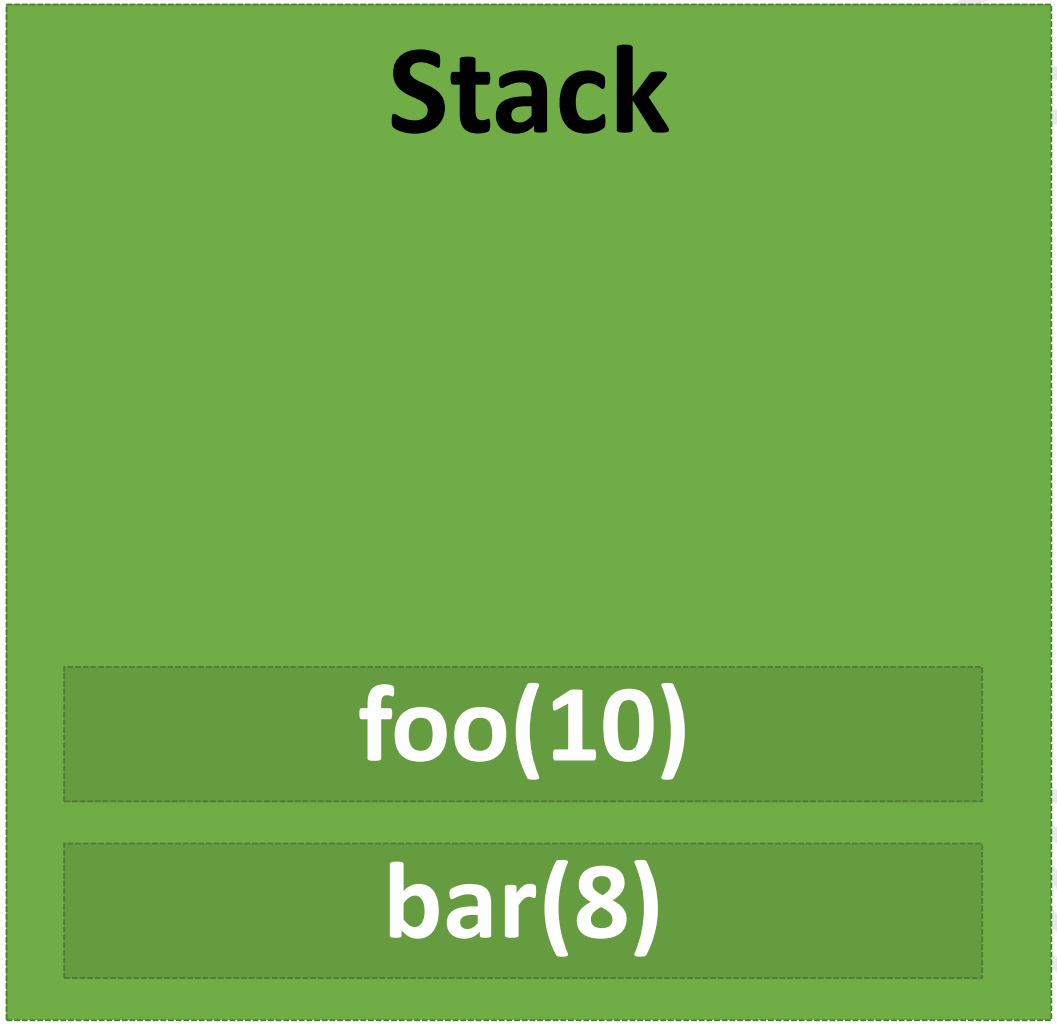
Stack calls

```
function foo(x) {
    return x * x;
}
function bar(y) {
    return foo(y + 2);
}

bar(8);
```



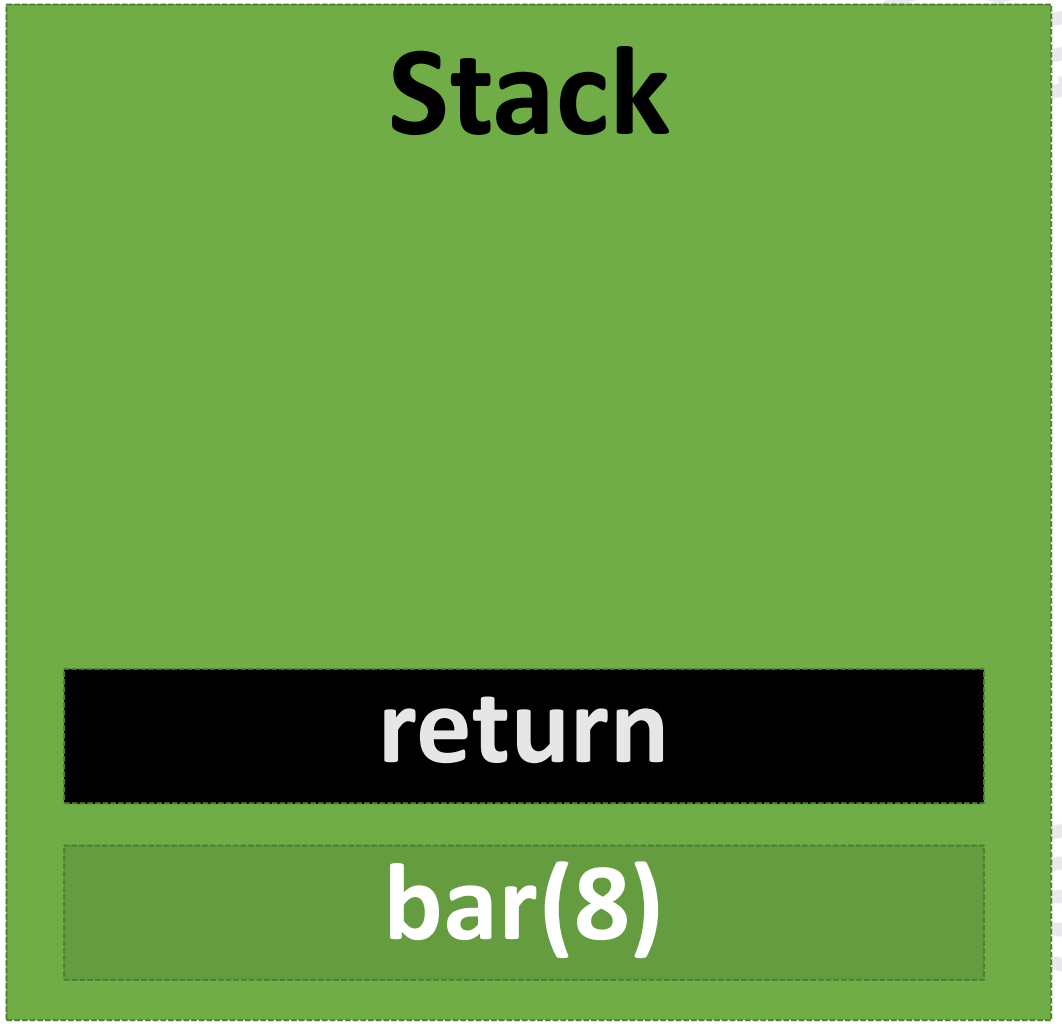
Stack calls

```
function foo(x) {
    return x * x;
}
function bar(y) {
    return foo(y + 2);
}

bar(8);
```



Stack calls

```
function foo(x) {
    return x * x;
}
function bar(y) {
    return foo(y + 2);
}

bar(8);
```

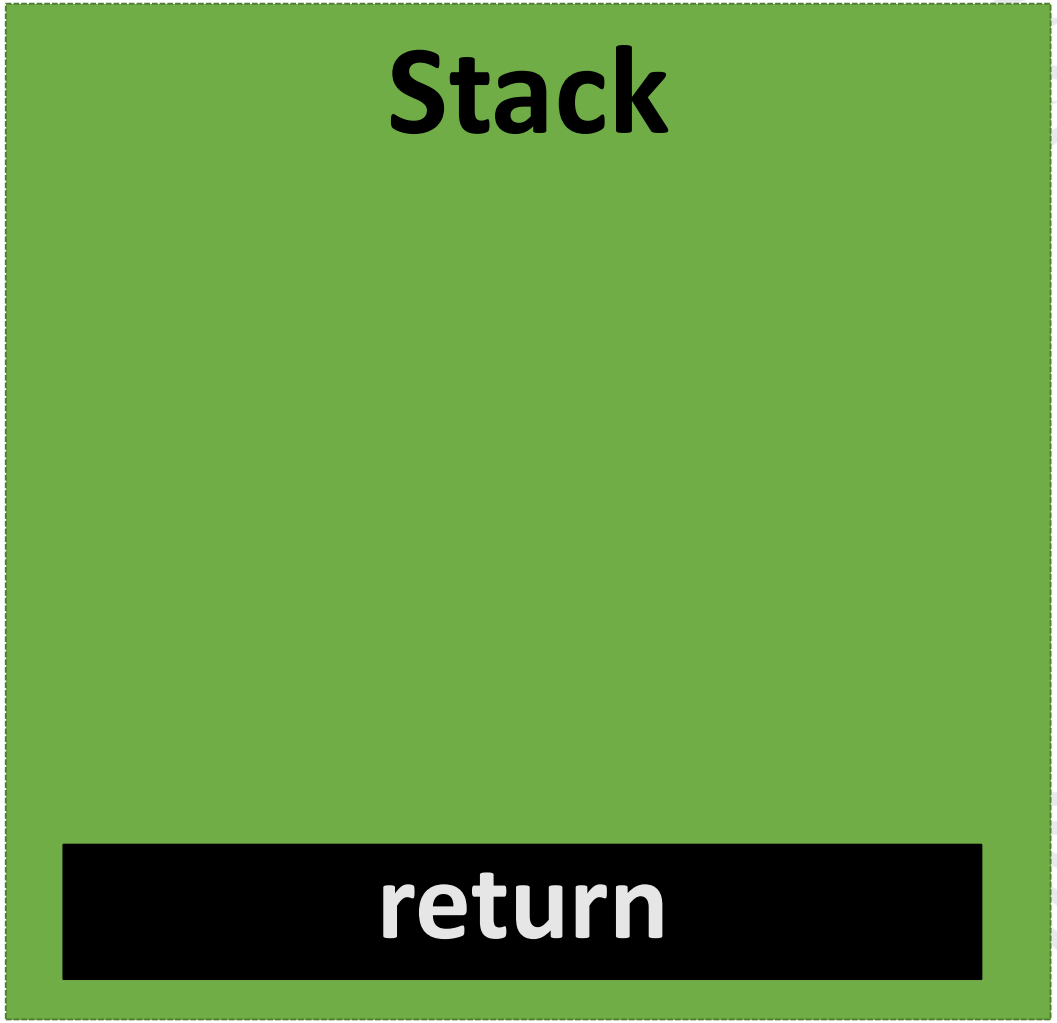


Stack calls

```
function foo(x) {
    return x * x;
}
function bar(y) {
    return foo(y + 2);
}

bar(8);
```

Stack calls

```
function foo(x) {
    return x * x;
}
function bar(y) {
    return foo(y + 2);
}

bar(8);
```

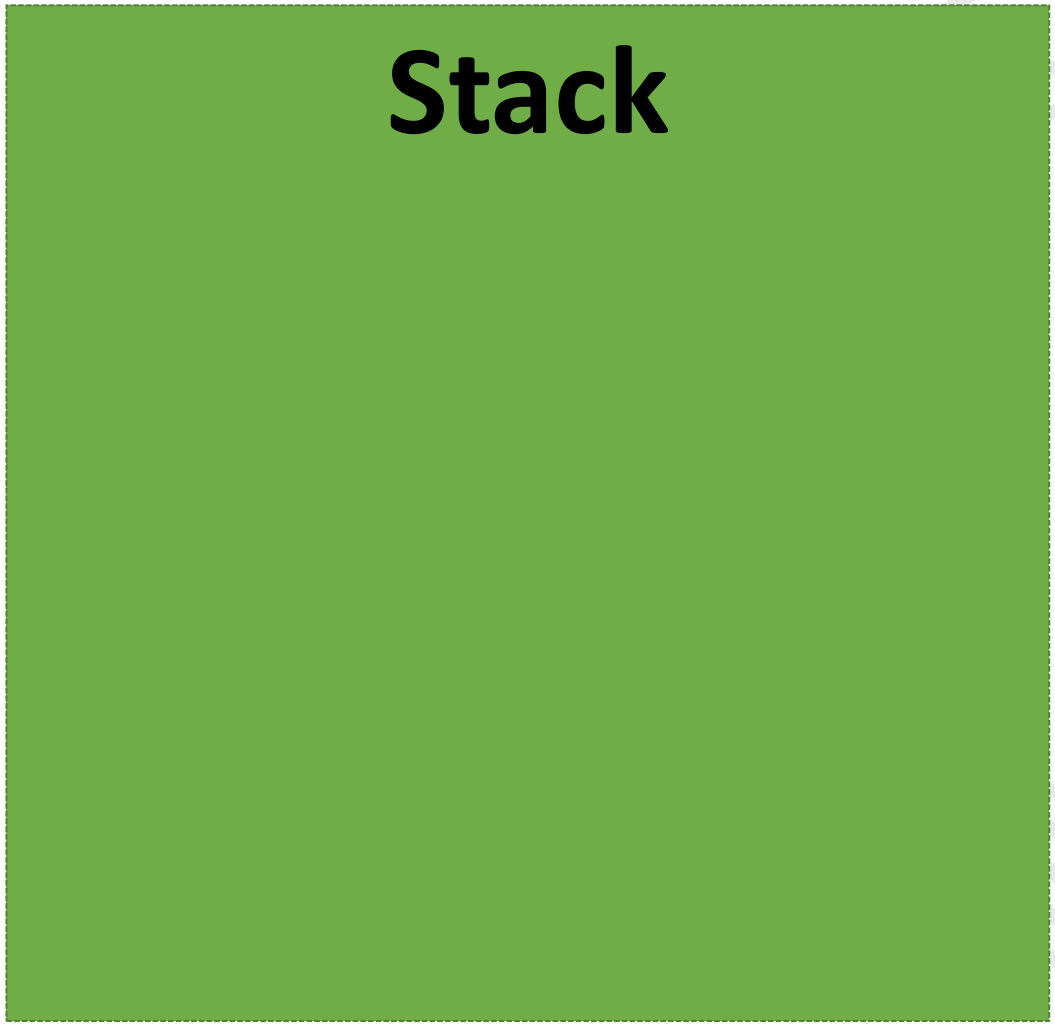


Stack calls

```
function foo(x) {
    return x * x;
}
function bar(y) {
    return foo(y + 2);
}

bar(8);
```

Stack calls

```
function foo(x) {
    return x * x;
}
function bar(y) {
    return foo(y + 2);
}

bar(8);
```



```
function init(el){
    el.addEventListener(
        "click",
        handler
    );
}
```

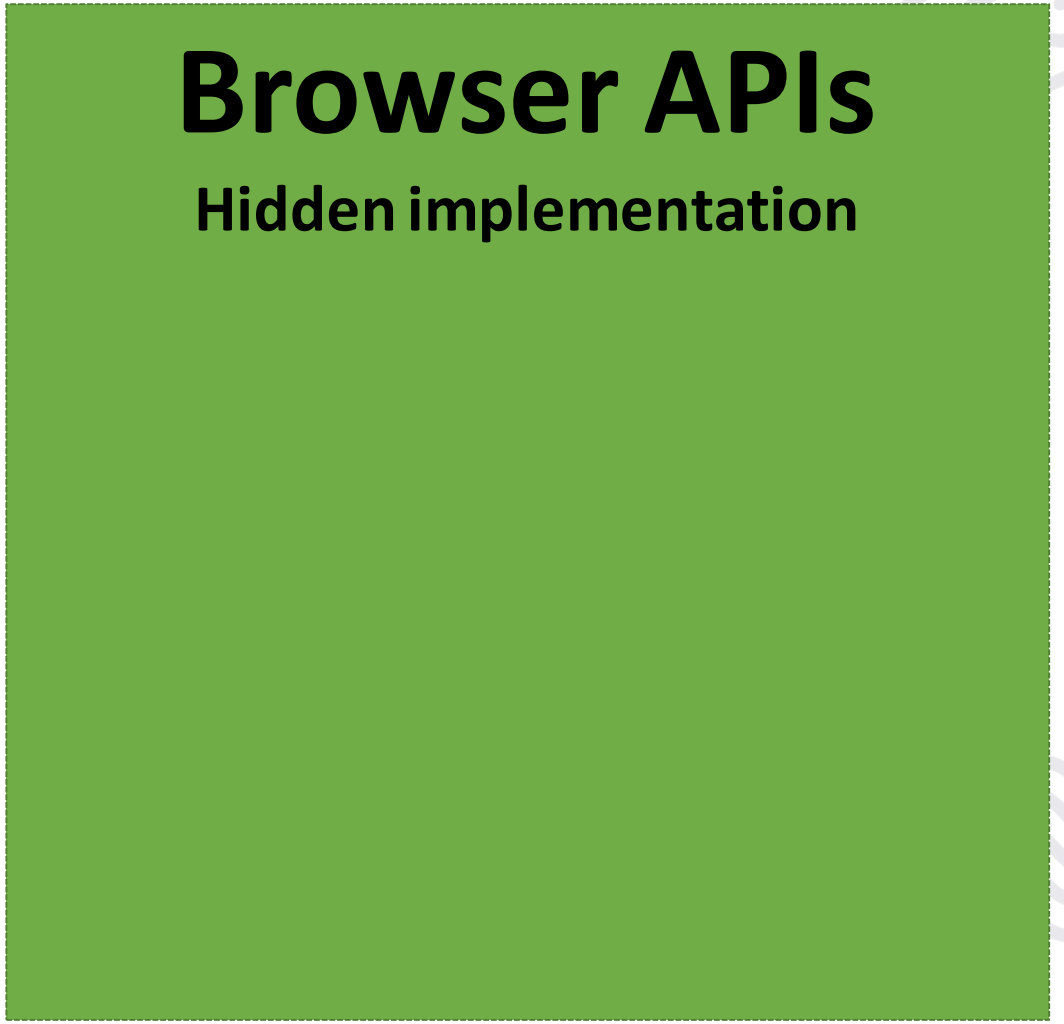
Stack calls

Stack

Browser APIs

Hidden implementation

Stack calls



Stack calls

Stack

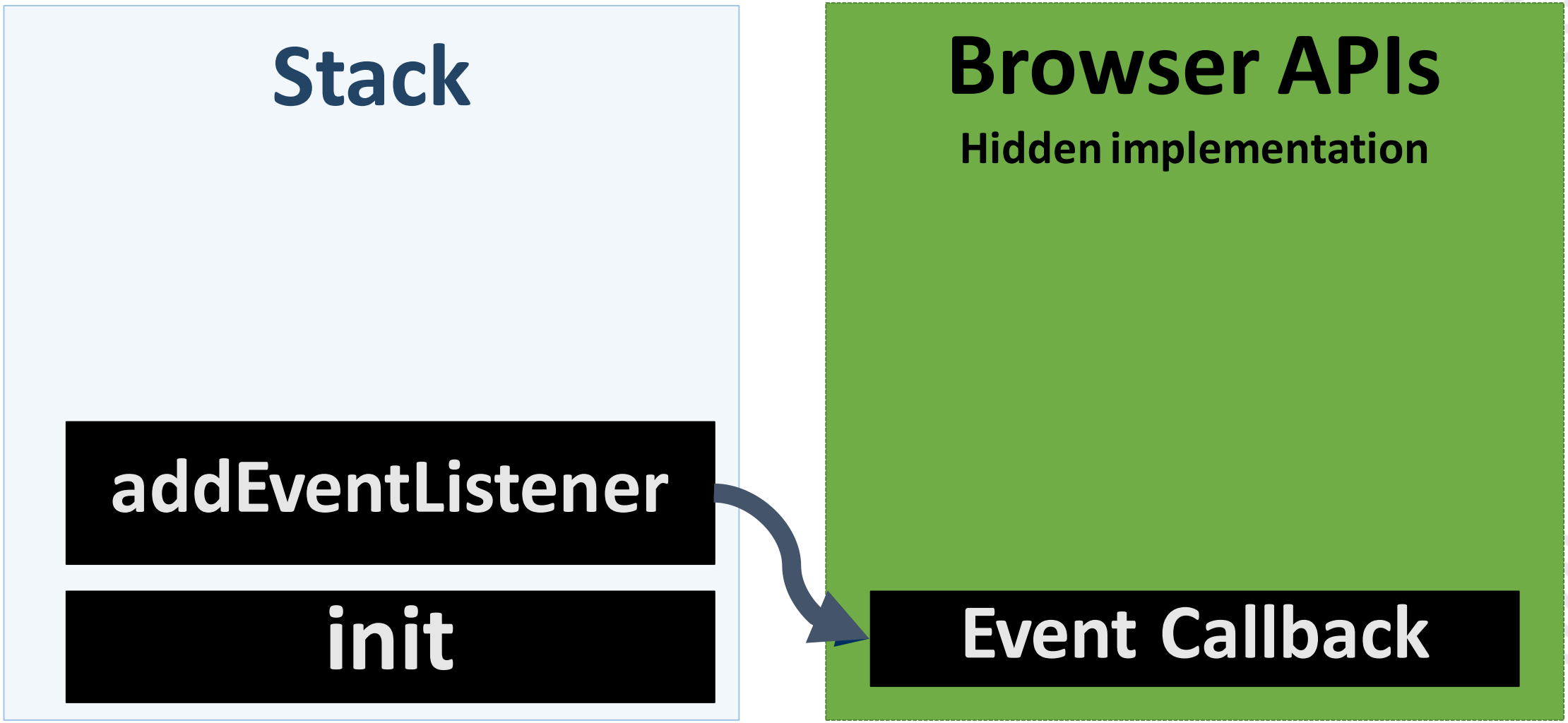
addEventListener

init

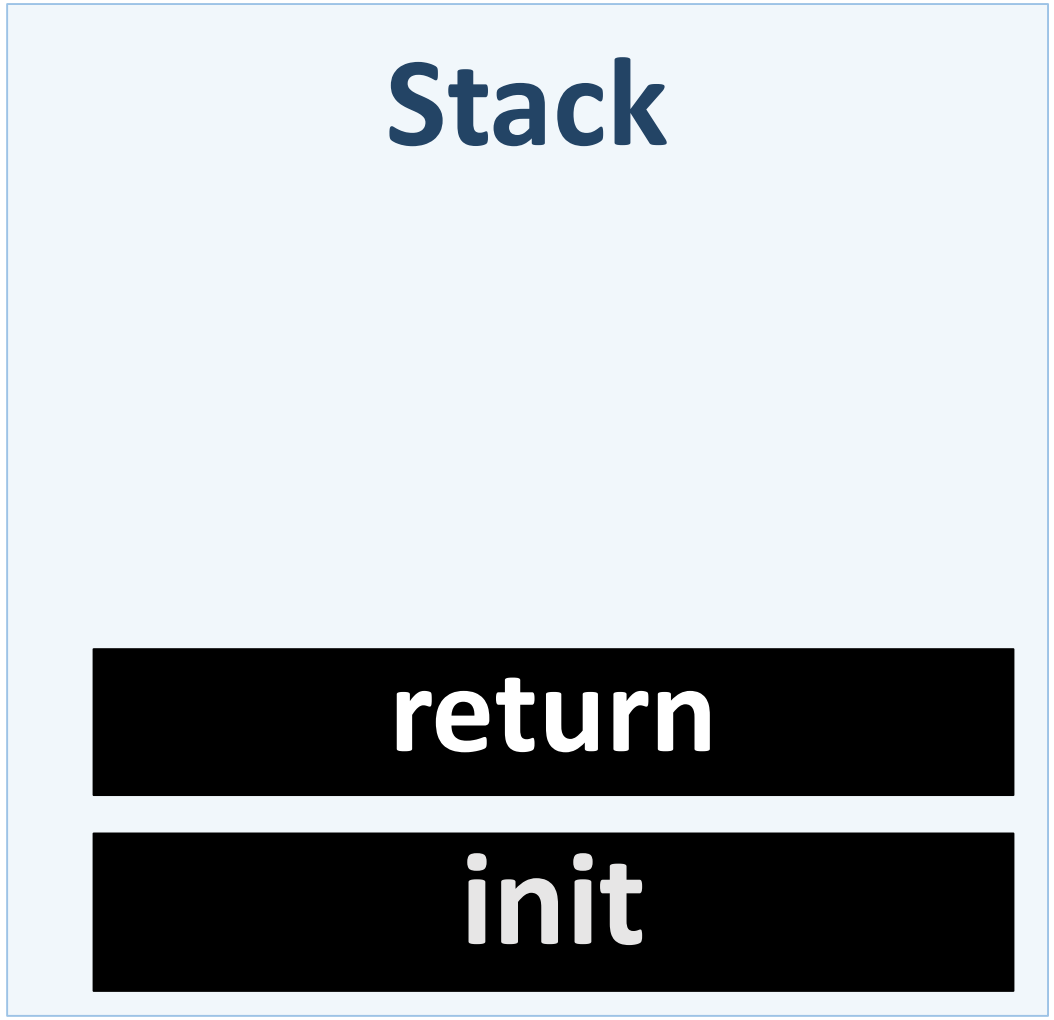
Browser APIs

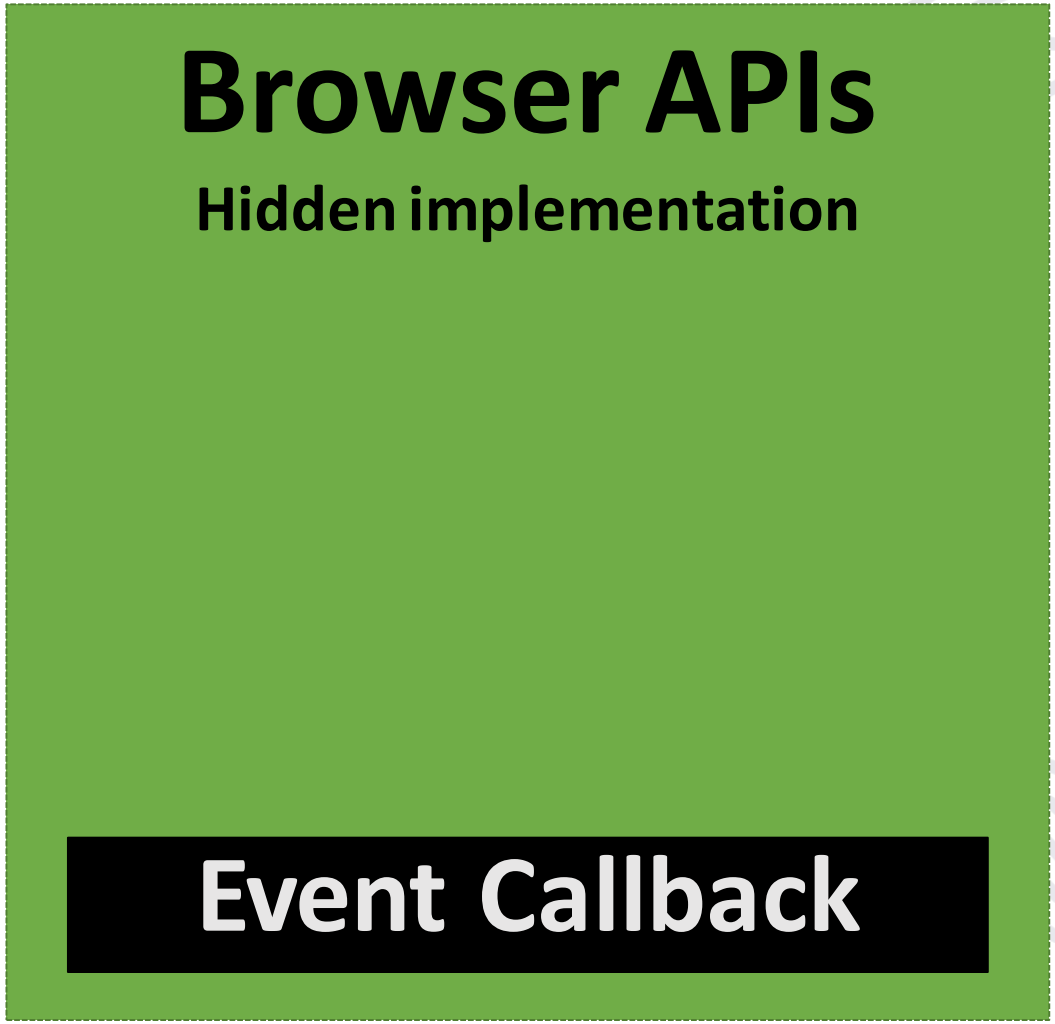
Hidden implementation

Stack calls

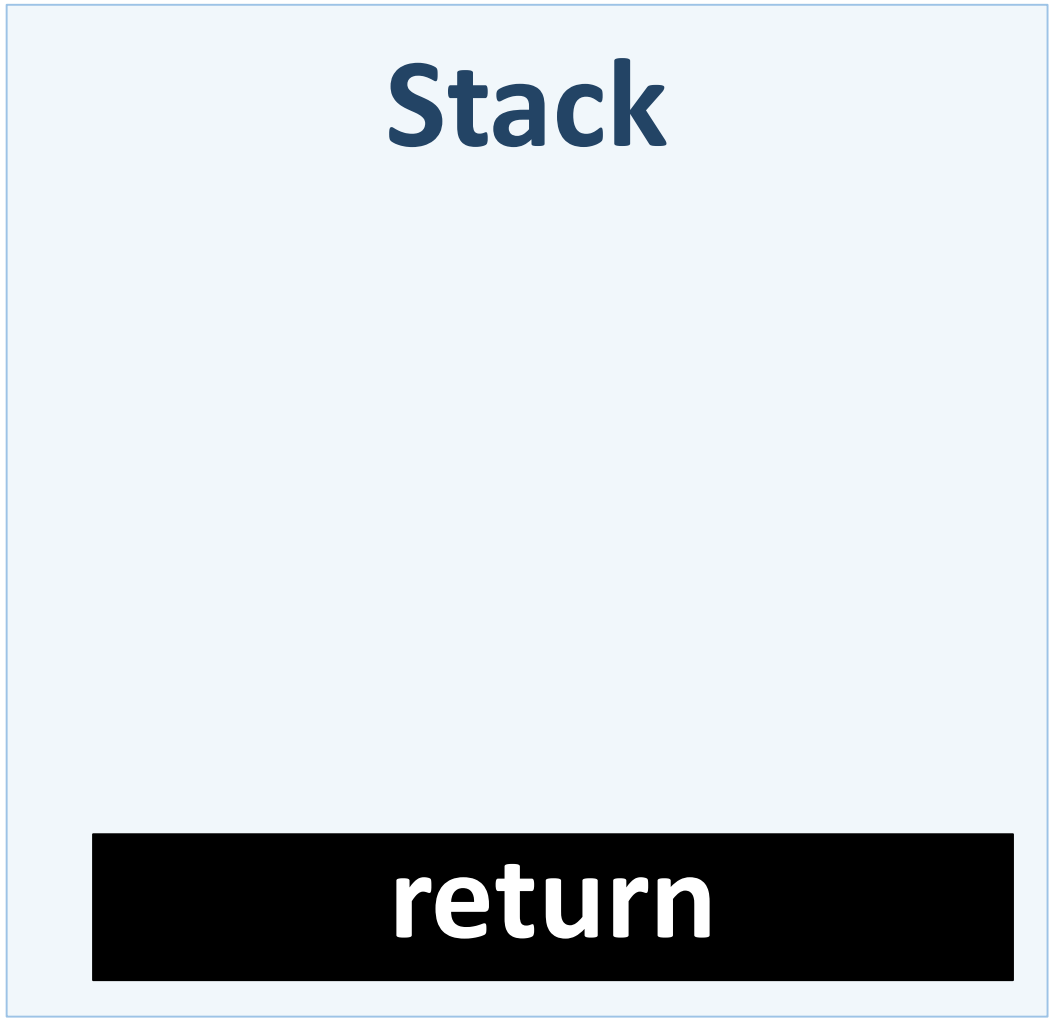

Stack
addEventListener
init
Browser APIs
Hidden implementation
Event Callback

Stack calls


Stack
return
init


Browser APIs
Hidden implementation
Event Callback

Stack calls


Stack
return


Browser APIs
Hidden implementation
Event Callback

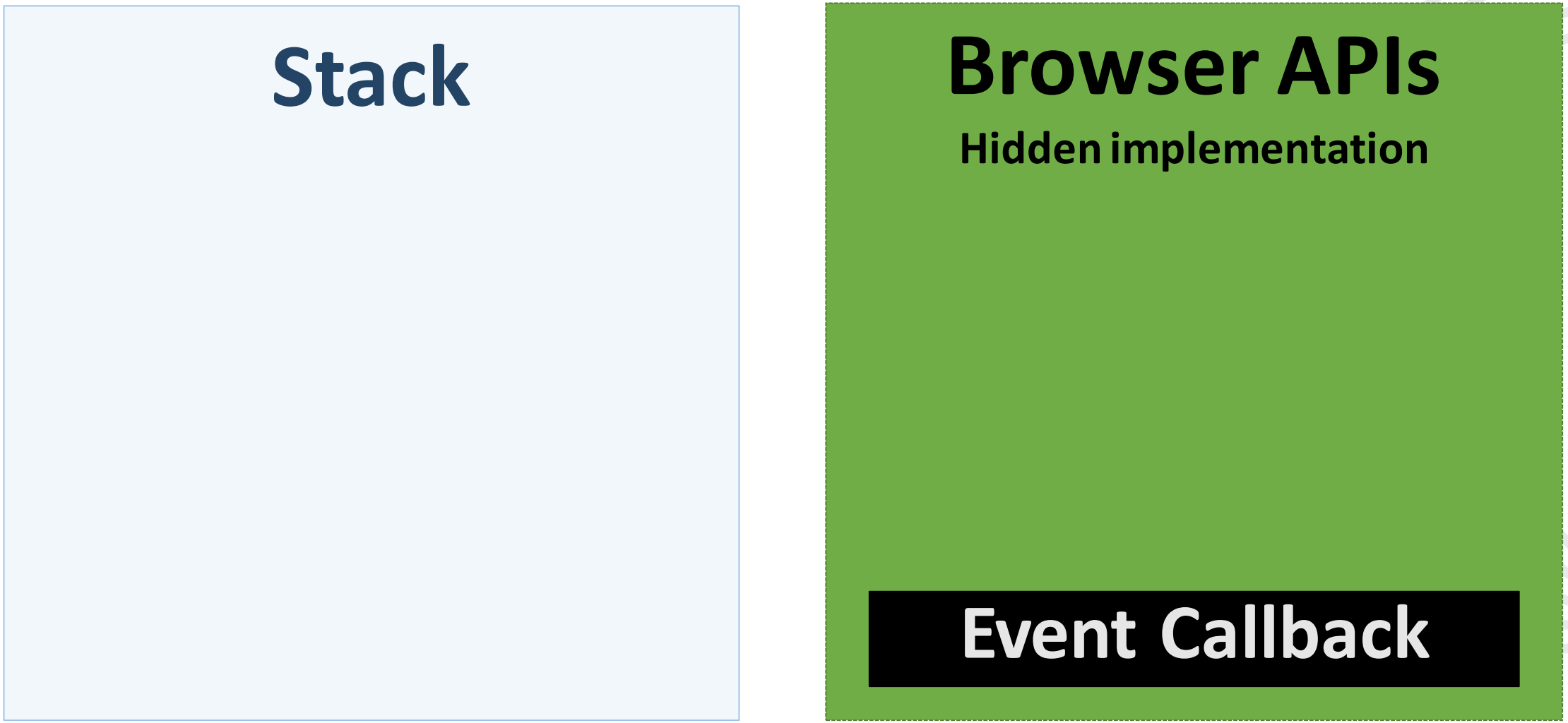
Stack calls

Browser APIs

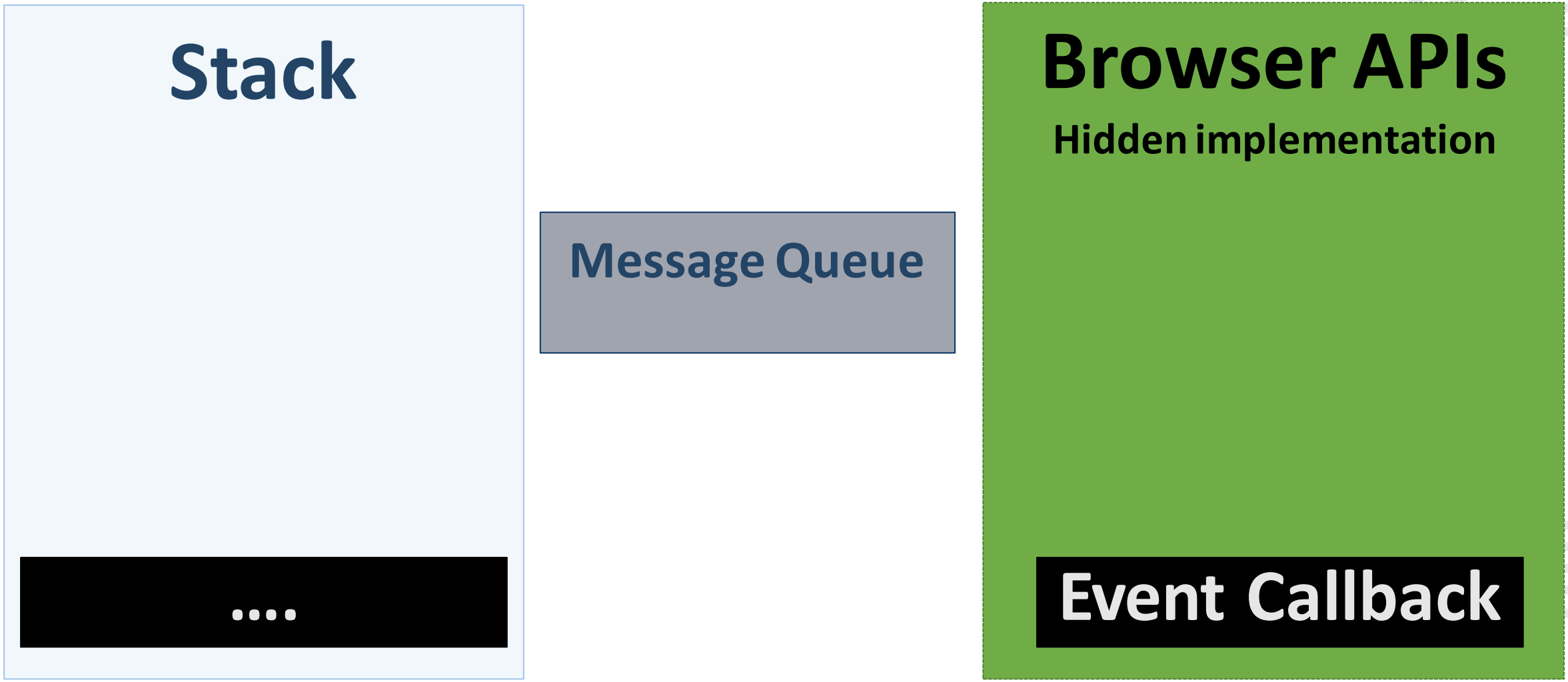
Hidden implementation

Event Callback

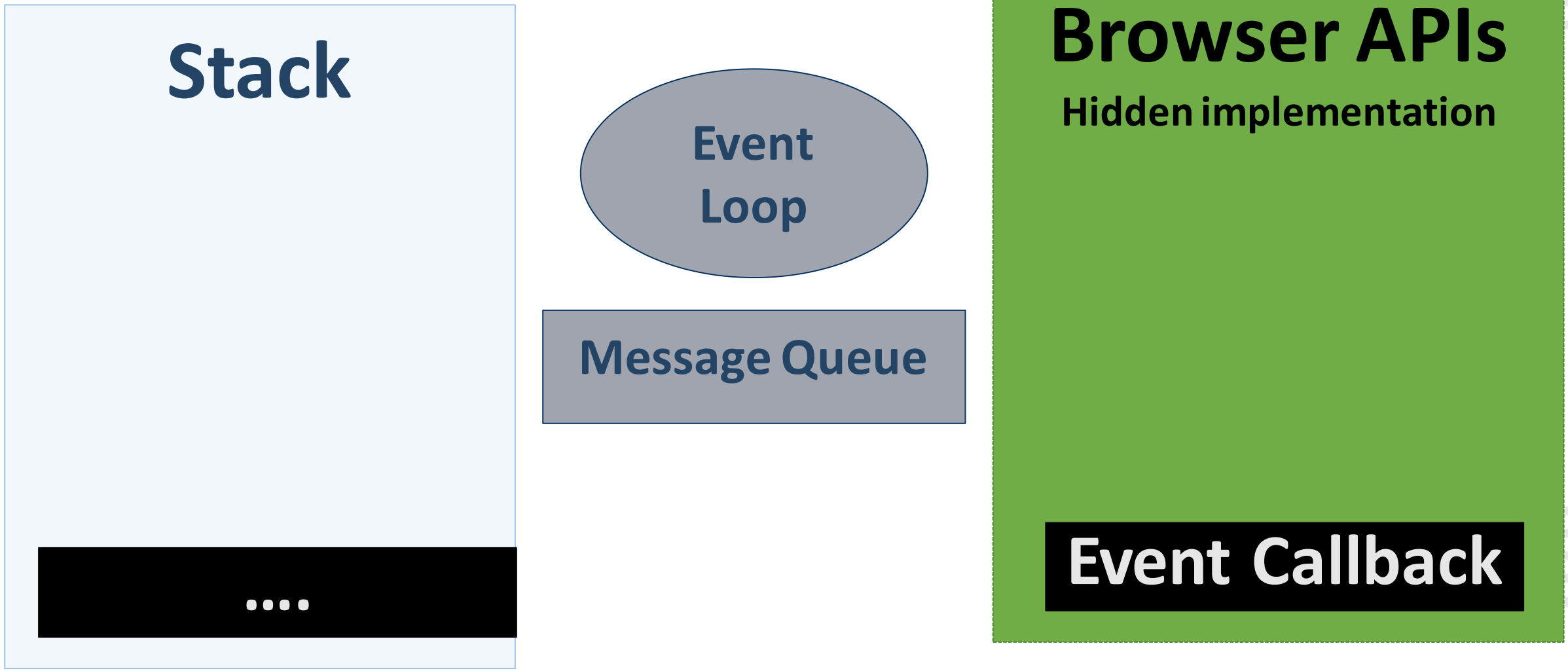
Stack calls



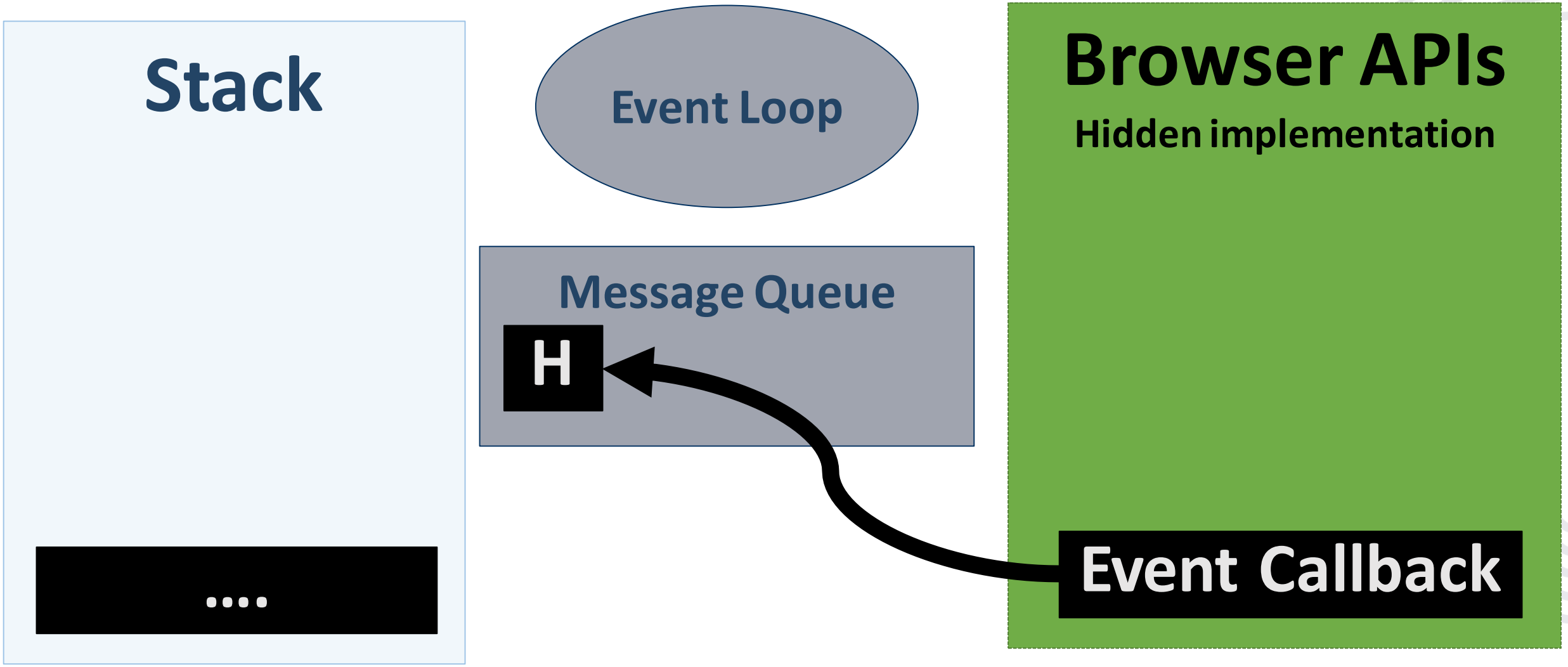
Stack calls


Stack
....
Message Queue
Browser APIs
Hidden implementation
Event Callback

Stack calls

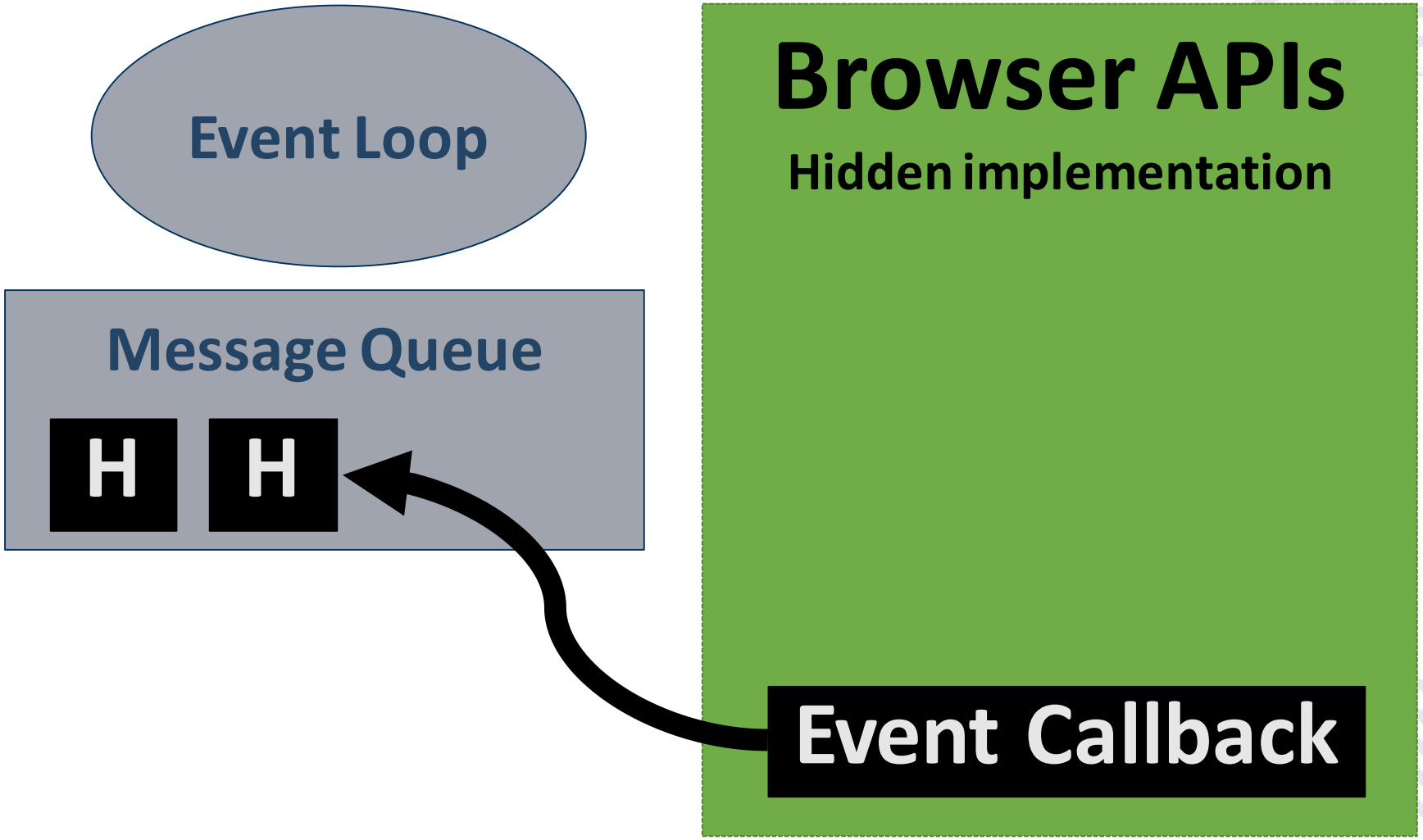

Stack
....
Event
Loop
Message Queue
Browser APIs
Hidden implementation
Event Callback

Stack calls


Stack
....
Event Loop
Message Queue
H
Browser APIs
Hidden implementation
Event Callback

Stack calls

Stack



Stack calls

Stack

Event Loop

Message Queue

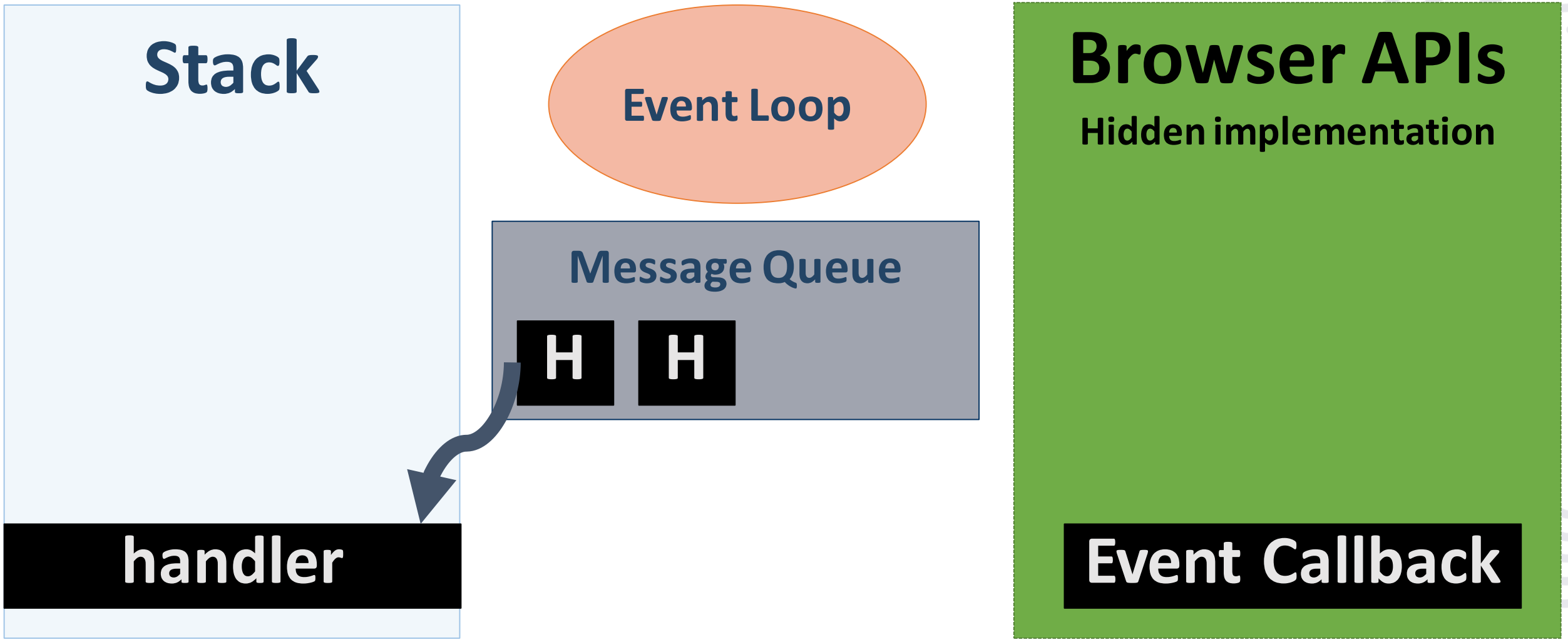
H H

Browser APIs

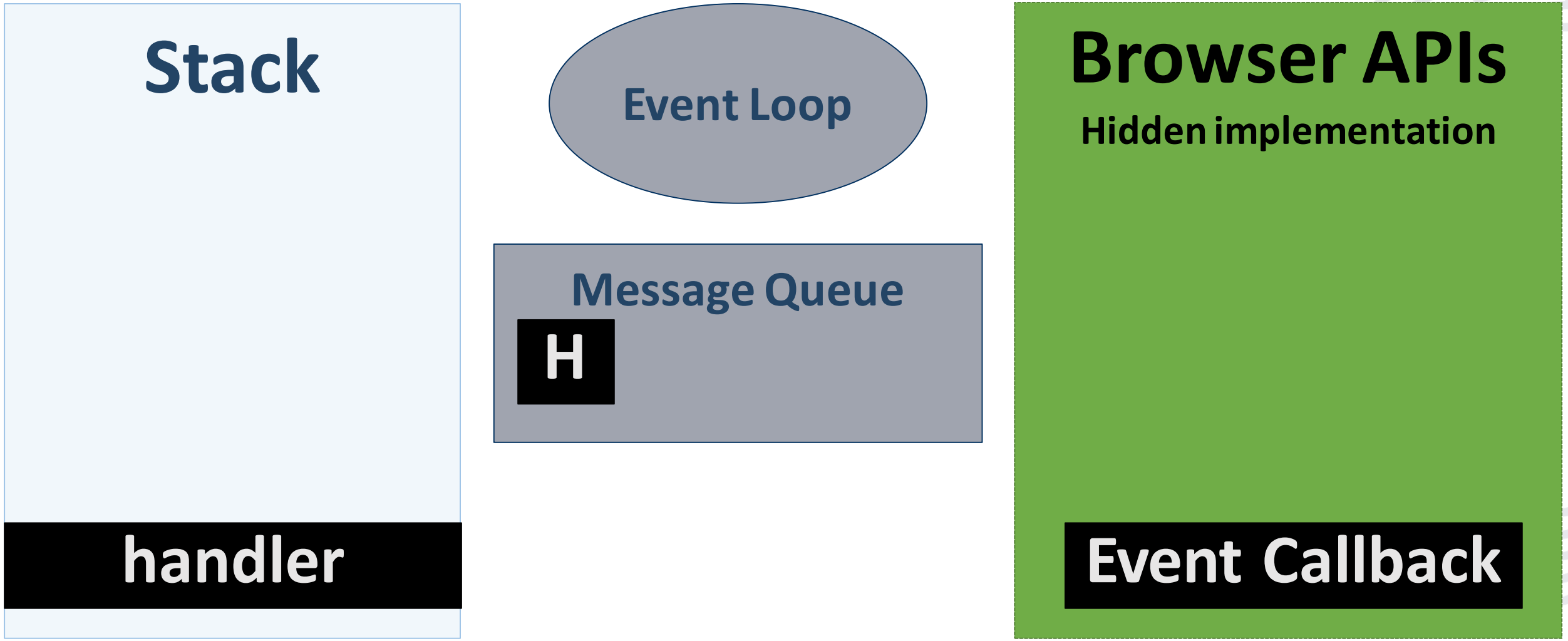
Hidden implementation

Event Callback

Stack calls


Stack
Event Loop
Browser APIs
Hidden implementation
Message Queue
H
H
handler
Event Callback

Stack calls


Stack
handler
Event Loop
Message Queue
H
Browser APIs
Hidden implementation
Event Callback

Stack calls

Stack

return

Event Loop

Message Queue

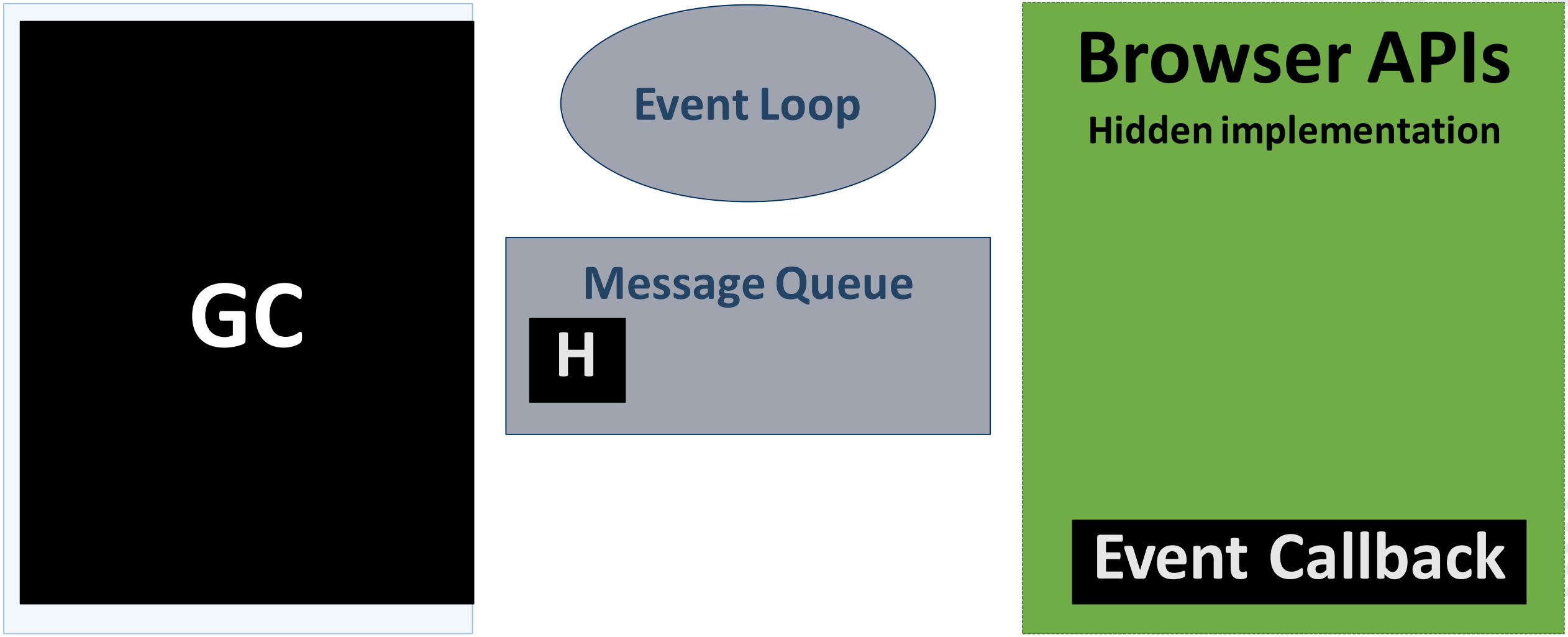
H

Browser APIs

Hidden implementation

Event Callback

Stack calls


GC
Event Loop
Message Queue
H
Browser APIs
Hidden implementation
Event Callback

Stack calls

Stack

Event Loop

Message Queue

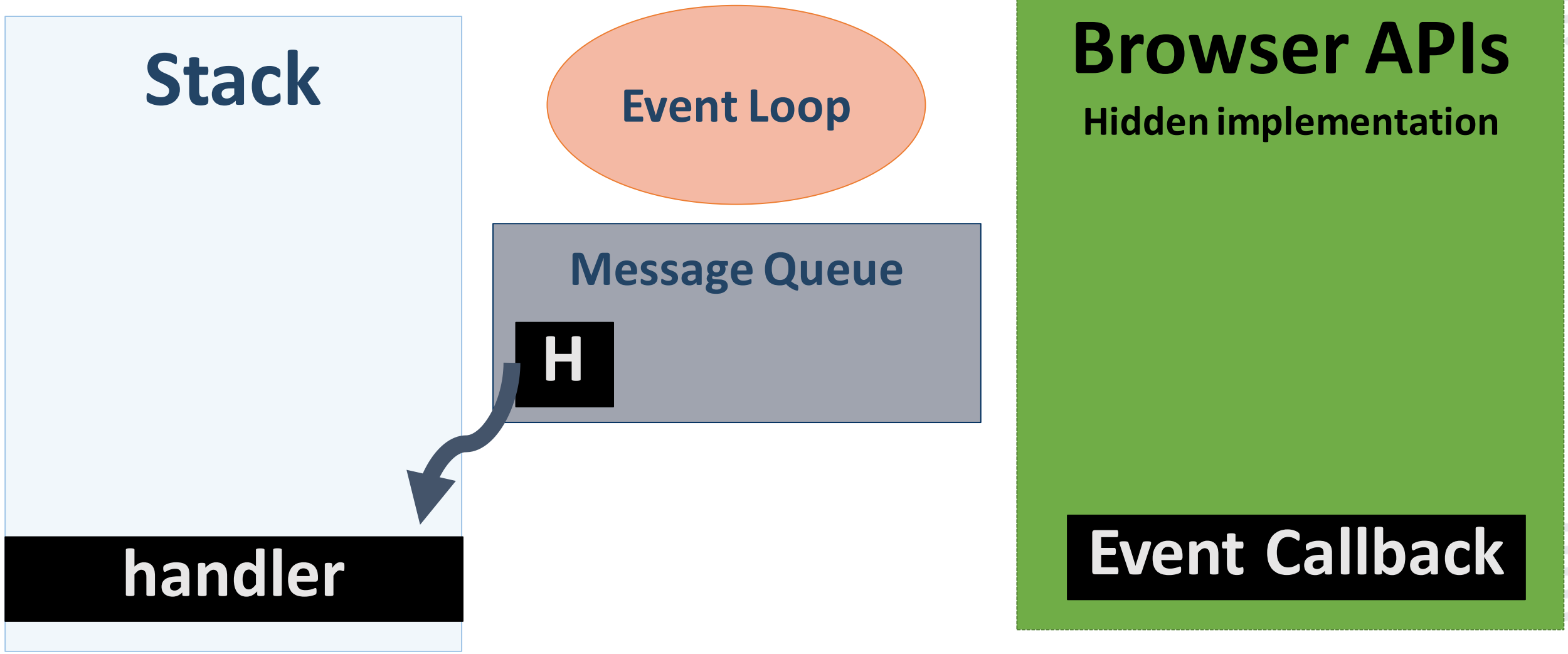
H

Browser APIs

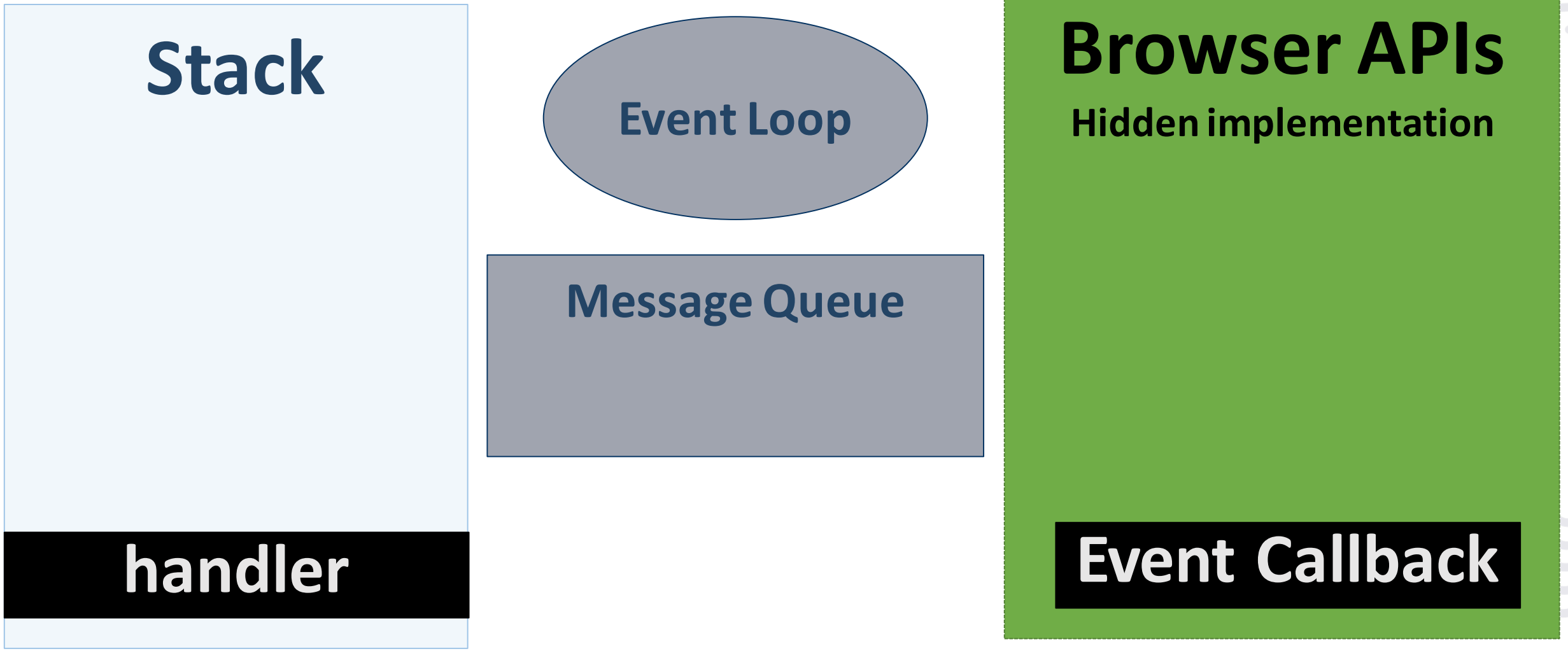
Hidden implementation

Event Callback

Stack calls


Stack
Event Loop
Browser APIs
Hidden implementation
Message Queue
H
handler
Event Callback

Stack calls


Stack
handler
Event Loop
Message Queue
Browser APIs
Hidden implementation
Event Callback

Stack calls

Stack

return

Event Loop

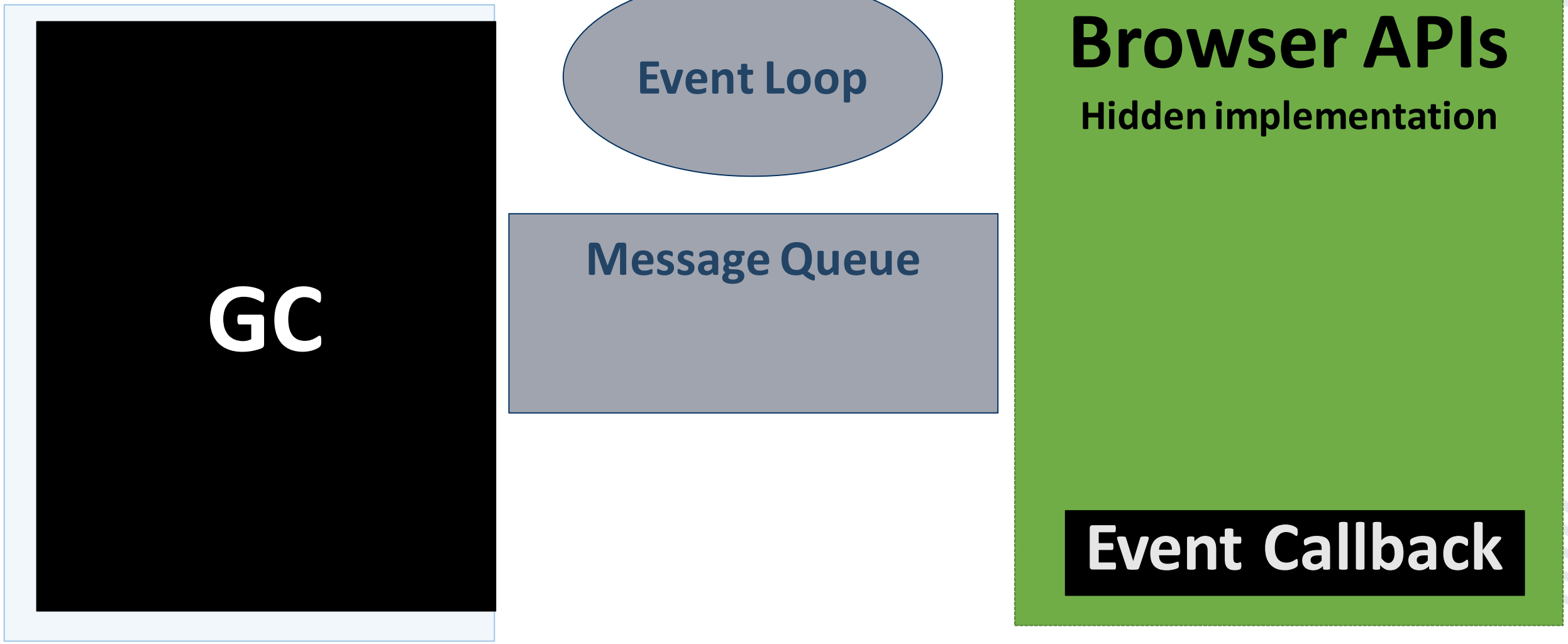
Message Queue

Browser APIs

Hidden implementation

Event Callback

Stack calls


GC
Event Loop
Message Queue
Browser APIs
Hidden implementation
Event Callback

Stack calls

Stack

Event Loop

Message Queue

Browser APIs

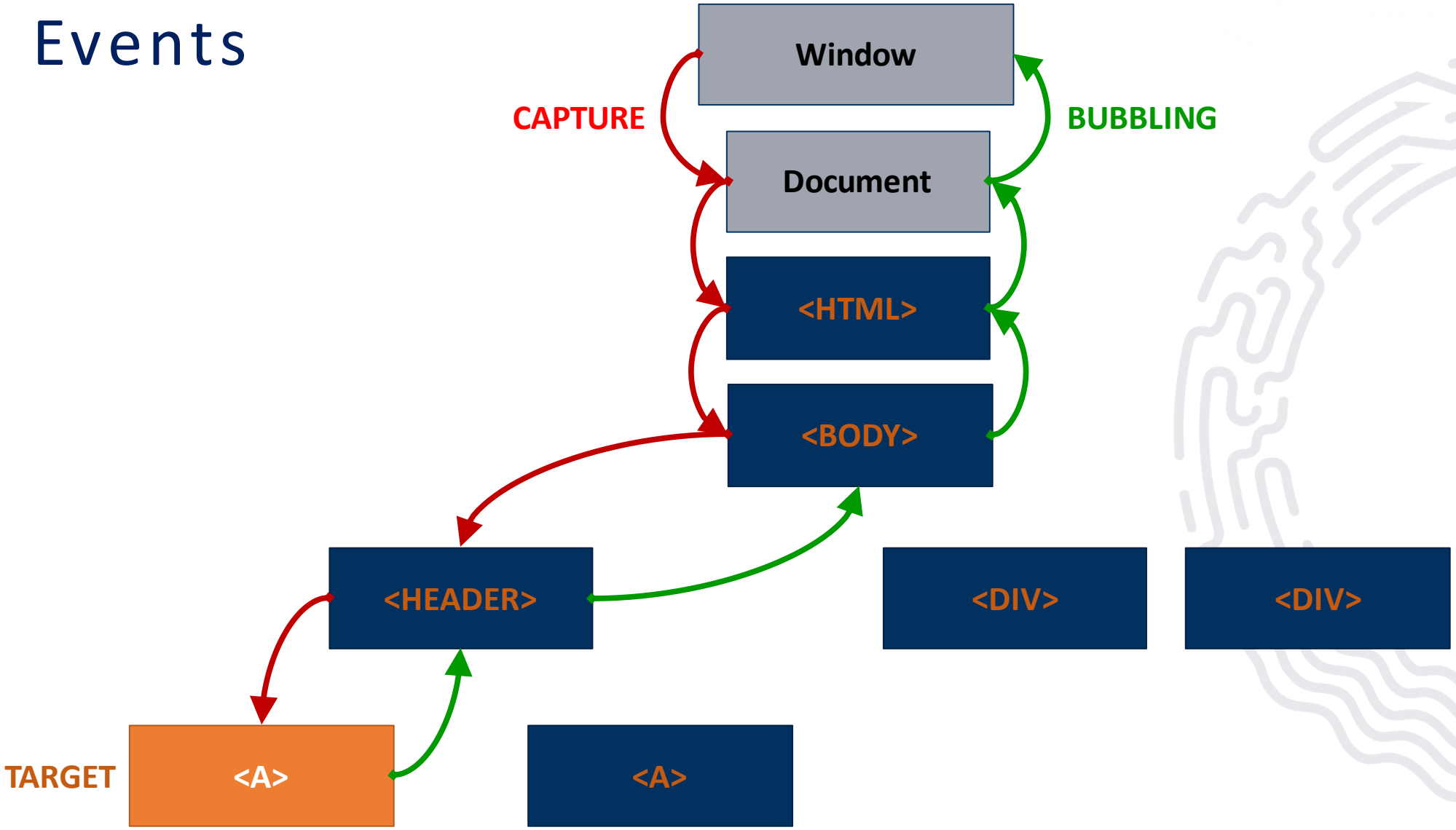
Hidden implementation

Event Callback

Handling DOM Events

DOM Events

DOM Events


Window
CAPTURE
BUBBLING
Document
<HTML>
<BODY>
<HEADER>
<DIV>
<DIV>
TARGET
<A>
<A>

Event Types in DOM API

- **Mouse** events

```
click
mouseover
mouseout
mousedown
mouseup
```

- **Keyboard** events

```
keydown
Keypress
keyup
```

- **Touch** events

```
touchstart
touchend
touchmove
touchcancel
```

- **Focus** events

```
focus (got focus)
blur (lost focus)
```

- **DOM / UI** events

```
load
unload
resize
dragstart / drop
```

- **Form** events

```
input
change
submit
reset
```

Event Object

- Calls its **associated function**
- Passes a **single argument** to the function - a **reference** to the event object
- Contains a **number of properties** that describe the event that occurred

Event Object Properties and Methods

- **Properties**
  - target
  - timeStamp
  - isTrusted
  - clientX/Y
- **Methods**
  - preventDefault
  - stopPropagation
  - stopImmediatePropagation

Live Exercises

Event Handling

Event Handler

- Event registration is done by providing a **callback function**
- Three ways to register for an event:
  - With **HTML Attributes**
  - Using **DOM element properties**
  - Using **DOM event handler**

```
function handler(event){
    this => object, html reference
    event => object, event configuration
}
```

Event Listener

- addEventListener();

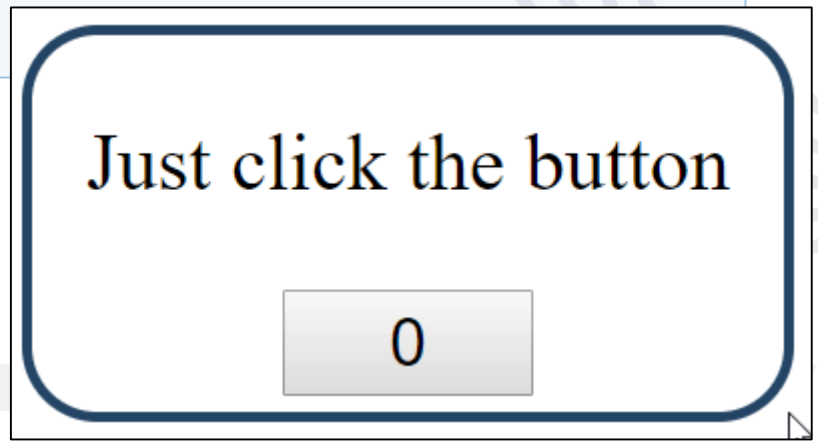
```
htmlRef.addEventListener( 'click' , handler , false );
```

- removeEventListener();

```
htmlRef.removeEventListener( 'click' , handler);
```

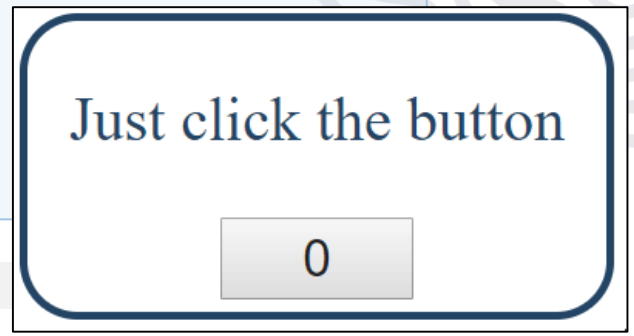
Attaching Click Event

```
const button = document.getElementsByTagName('button')[0];

button.addEventListener('click', clickMe);

function clickMe(e) {
  const target = e.currentTarget;
  const targetText = target.textContent;
  target.textContent = (+targetText) + 1;
}
```



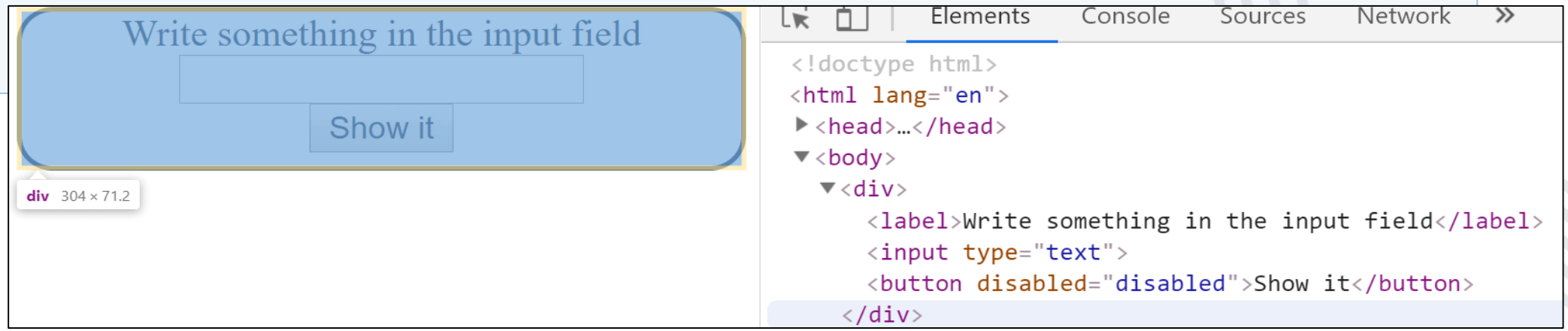
Attaching Hover Event

```
const button = document.getElementsByTagName('div')[0];
button.addEventListener('mouseover', function (e) {
    const style = e.currentTarget;
    const { backgroundColor } = style;

    if(backgroundColor === 'white'){
        targetStyles.backgroundColor = '#234465';
        targetStyles.color = 'white';
    } else {
        targetStyles.backgroundColor = 'white';
        targetStyles.color = '#234465';
    }}
);
```



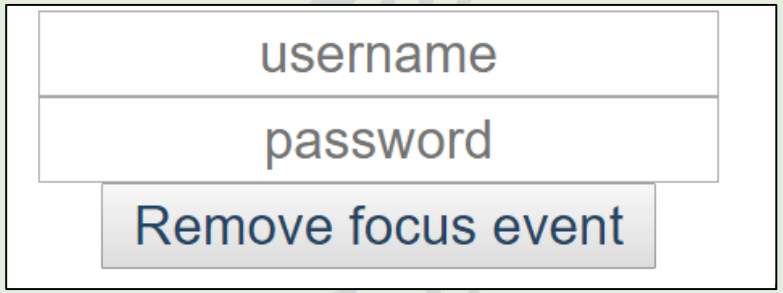
Attaching Input Event

```
const inputField = document.getElementsByTagName('input')[0
];
const button = document.getElementsByTagName('button')[0];

inputField.addEventListener('input', function () {
    button.setAttribute('disabled', 'false')
});
```



Remove Events

```
const password = document.querySelector('input[type="password"]');
const button = document.querySelector('button');
password.addEventListener('focus', focusEvent);

function focusEvent (){
    event.target.style.background = '#234465';
}

password.addEventListener('blur', (event) => {
    event.target.style.background = '';
});

button.addEventListener('click', () => {
    password.removeEventListener('focus', focusEvent);
});
```



Multiple Events

- The **`addEventListener()`** method also allows you to add many events to the same element, without overwriting existing events:

```
element.addEventListener("click", function);
element.addEventListener("click", myFunction);
element.addEventListener("mouseover", mySecondFunction);
element.addEventListener("mouseout", myThirdFunction);
```

*Note that you don't use the "**on**" prefix for the event use "**click**" instead of "**onclick**"*

SetInterval() / ClearInterval()

- In JS we can start / stop timers (intervals)

```
let intervalID = setInterval(
  function() {
    console.log("1 sec. passed");
  },
  1000
); // Delay = 1000 ms = 1 second
```

- Remove (cancel) existing timer

```
clearInterval(intervalID); // Stop the timer
```

Live Exercises

Event Delegation

DOM Event Delegation

- Allows you to **avoid** adding event listeners to specific nodes
- Event listener is assigned to a **single ancestor**

```
<ul id="parent-list">
    <li id="post-1">Item 1</li>
    <li id="post-2">Item 2</li>
</ul>
```

```
document.getElementById("parent-list")
  .addEventListener("click", function(e) {
    if(e.target && e.target.nodeName == "LI") {
        console.log(
          "List item ", e.target.id.replace("post-", ""),
          " was clicked!");
    }
});
```

Pros and Cons

- **Benefits**
  - Simplifies initialization
  - Saves memory
  - Less code
- **Limitations**
  - Event must be bubbling
  - May add CPU load

Live Exercises

Summary

- Event Loop
- Event Types
- Event Object Properties and Methods
  - preventDefault
  - stopPropagation
- Handling Events
  - Attach
  - Remove

Questions?

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © Kingsland University – https://kingslanduniversity.com

KINGSLAND UNIVERSITY

THANK YOU