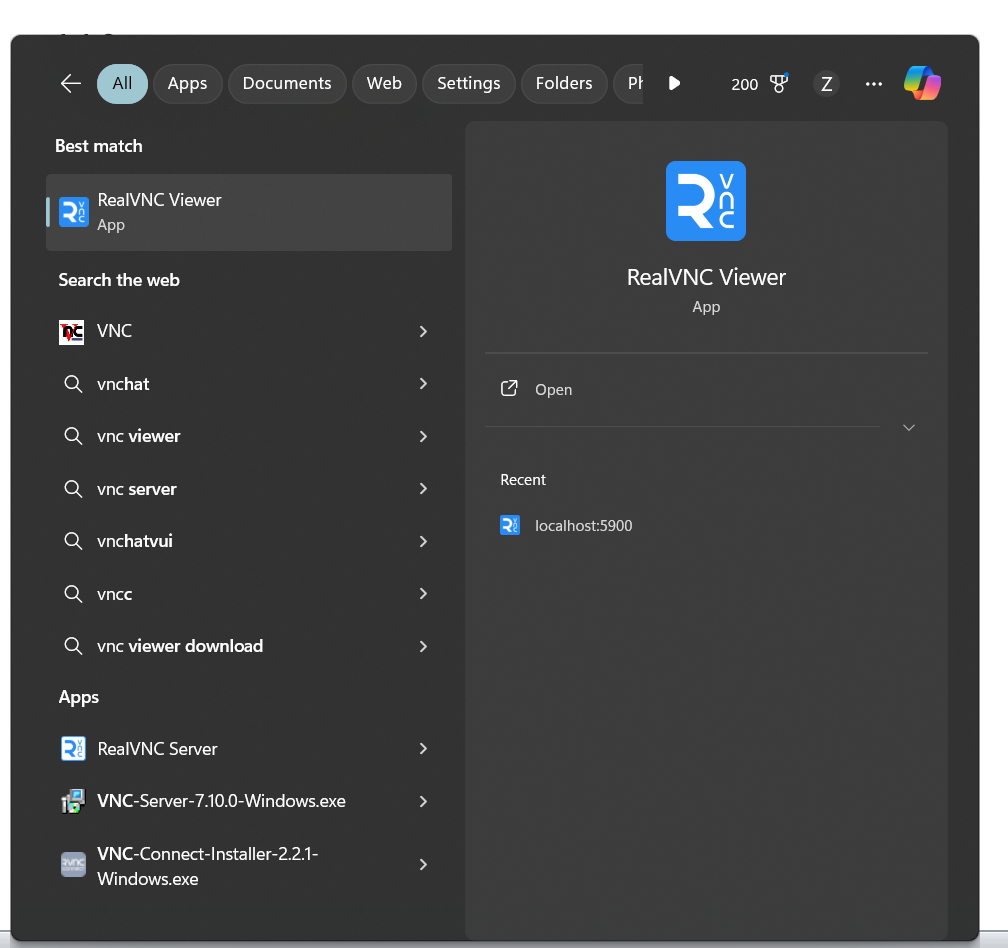
# LAB 05.1: Web Secruity – SOP ( Same Origin Policy )

**Tasks**

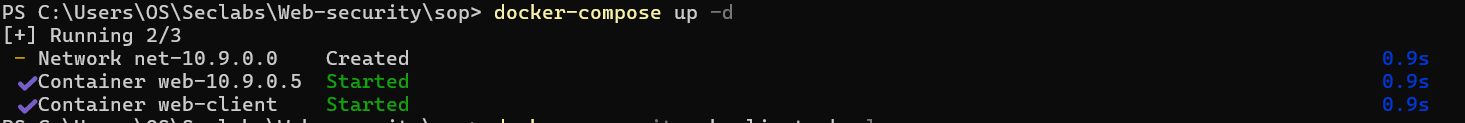
**1. SOP Lab**

**1.1 Setup**

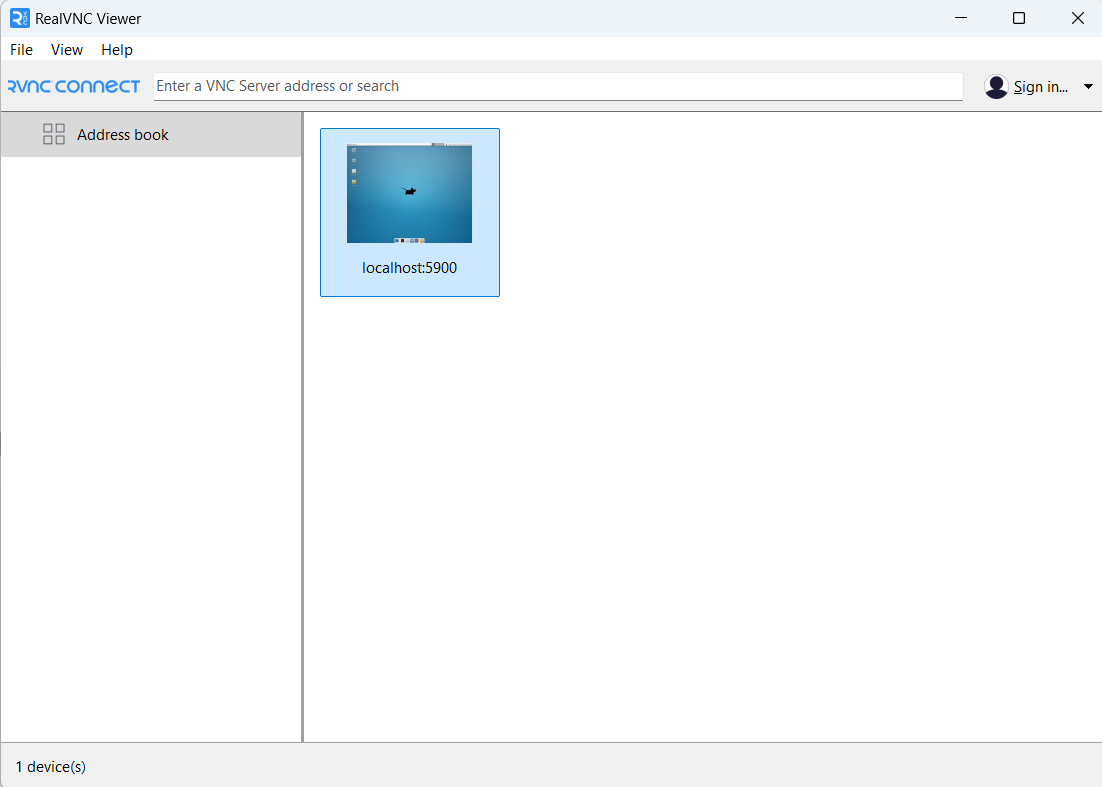
**a) Install VNC Viewer On the host computer:**



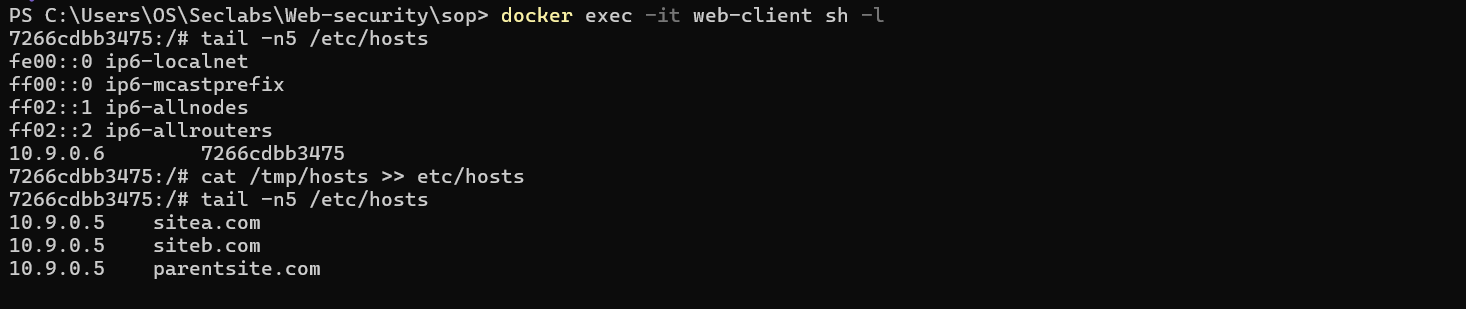
**b) Create docker container**



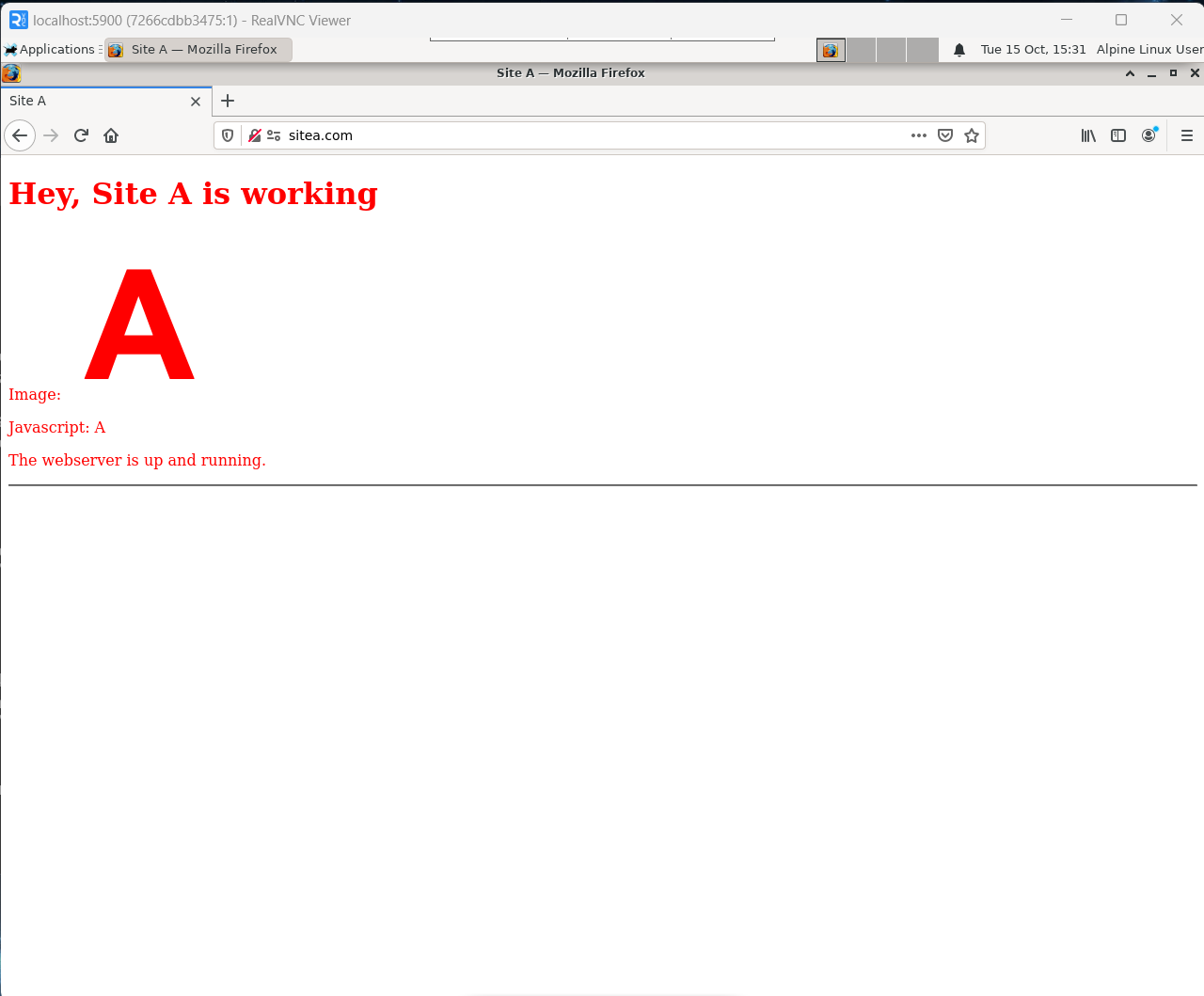
**c) Fire up VNC Viewer**



**d) Attach to the console of web-client docker container**



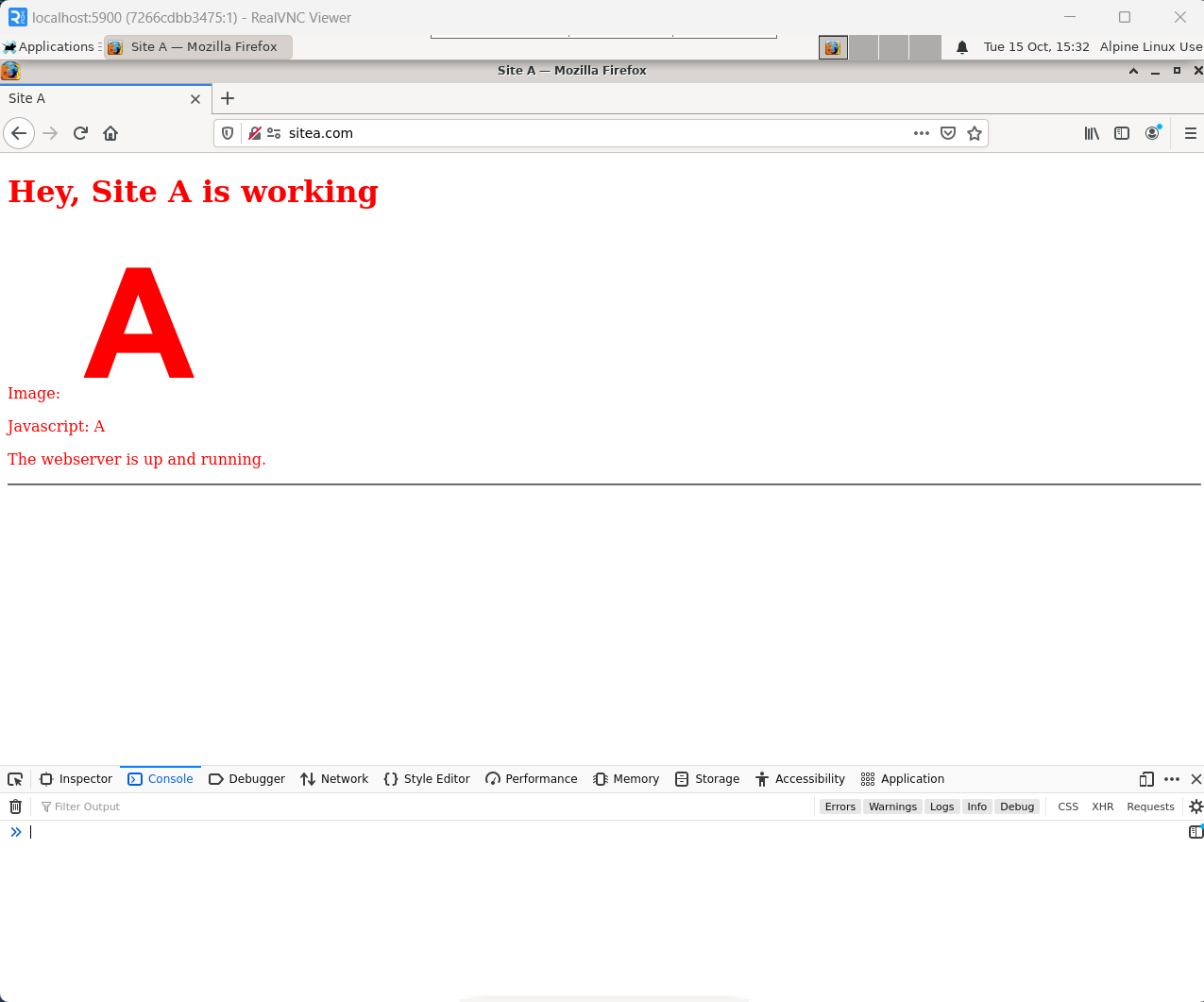
**e) Fire up website**



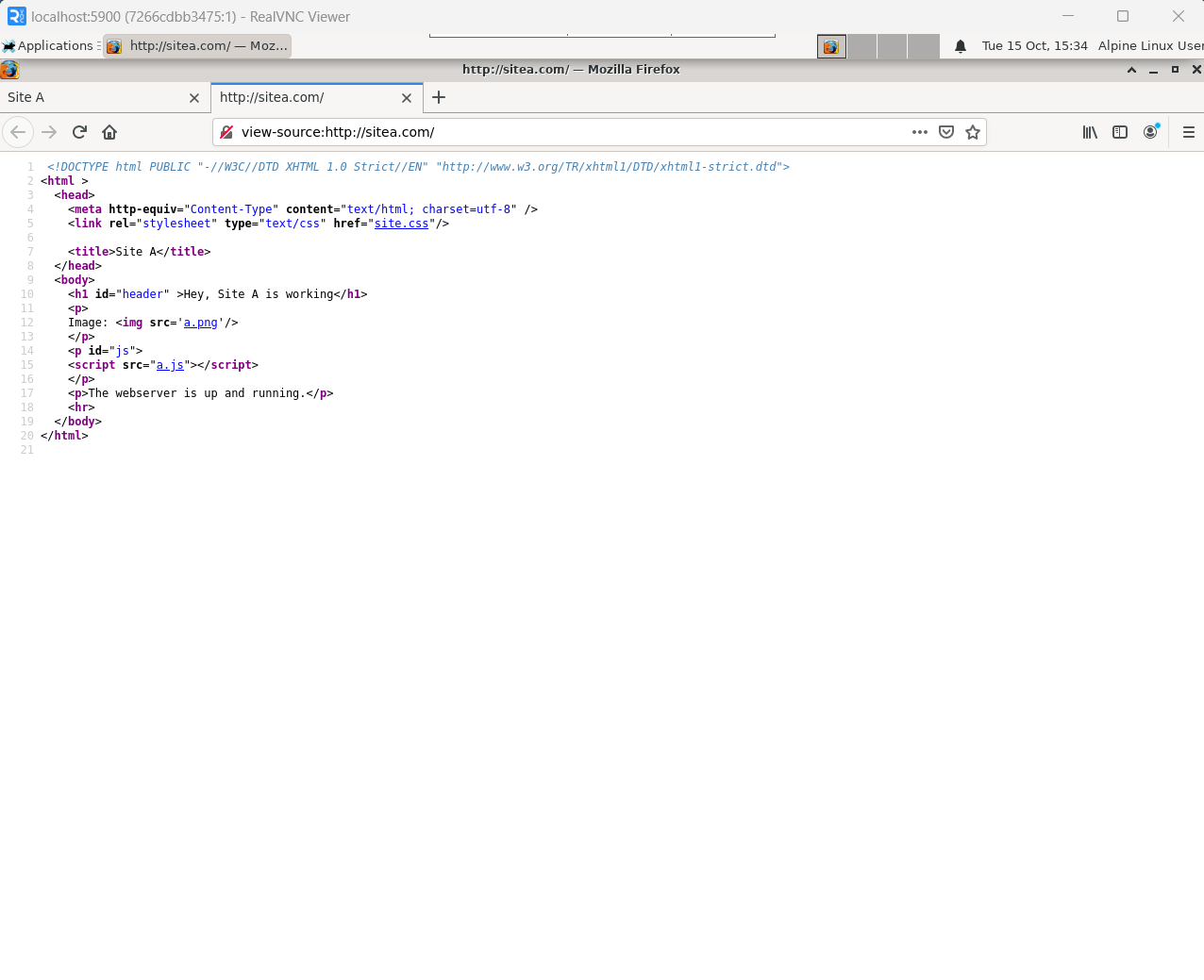
**1.2 Using JS to access DOM objects of the web site.**

**-Load sitea in browser**

**-Open Web Developer >> Console Panel ( Ctrl + Shift + K )**



**-Right-click on page -> View page source to display source of web page.**

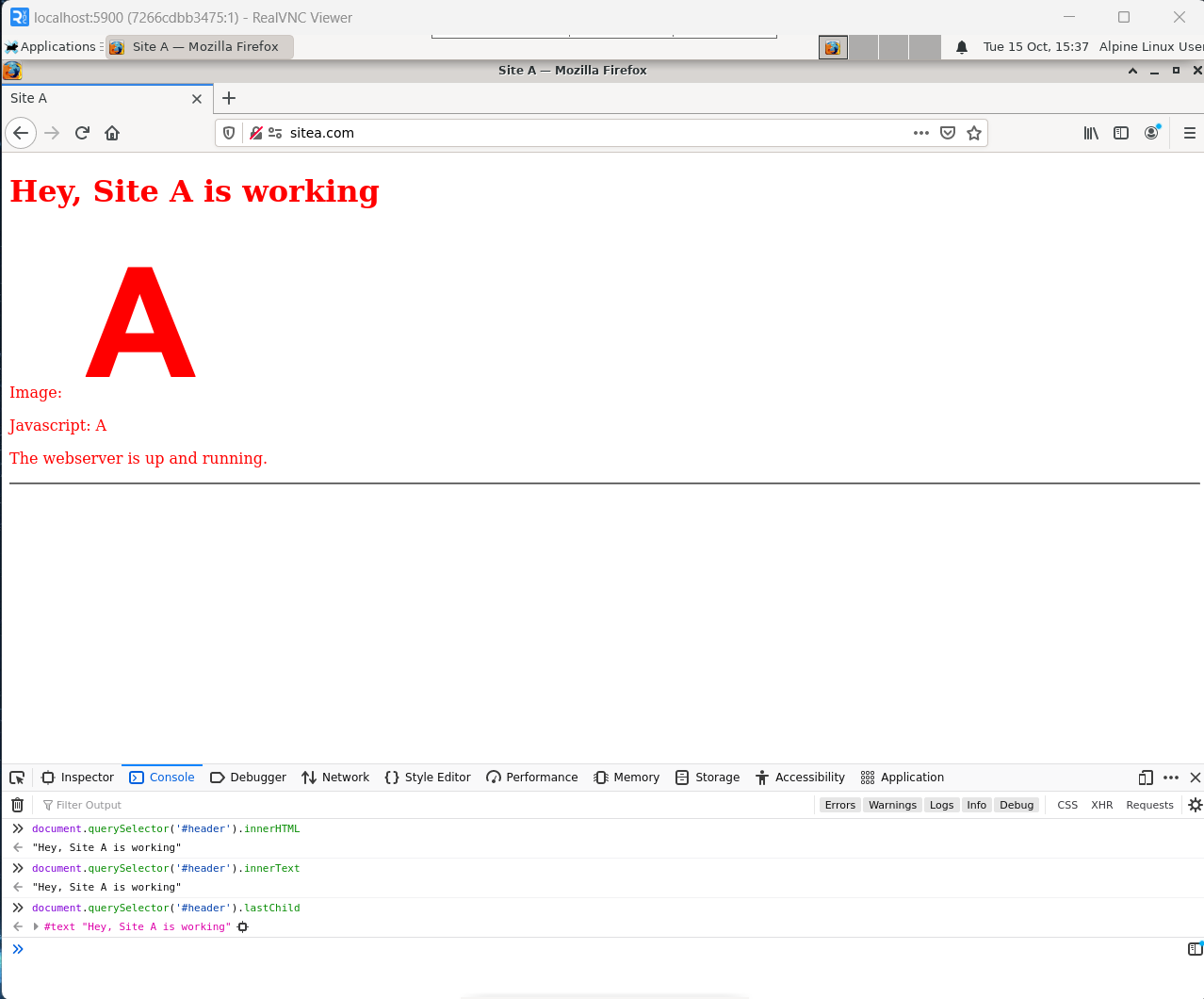


Ids list: header,js

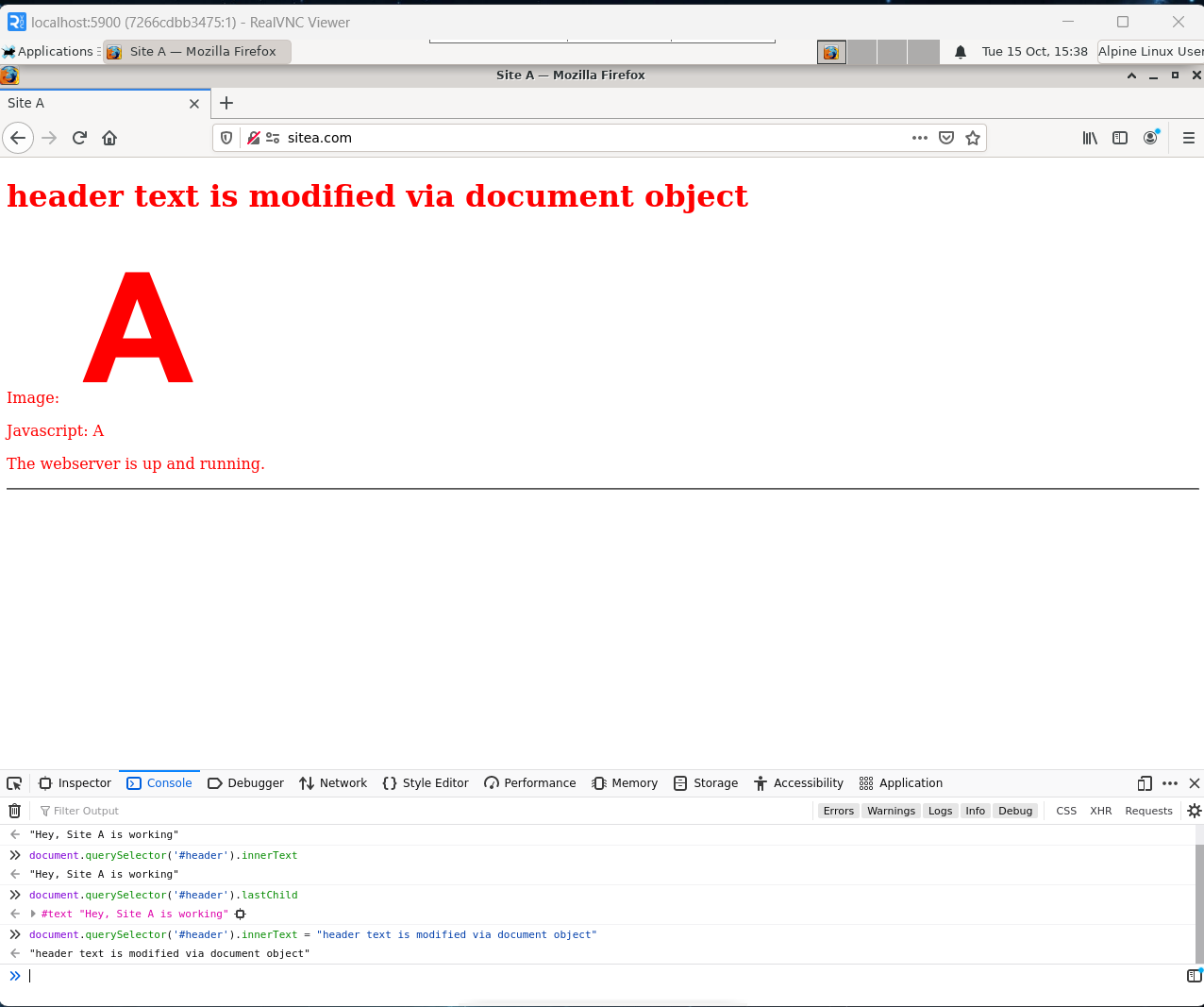
-Go to Console Panel, with javascript:

**a) Try to access DOM objects of page: document.body, document.body.innerText,**

**document.body.innerHTML or whichever that you can:**



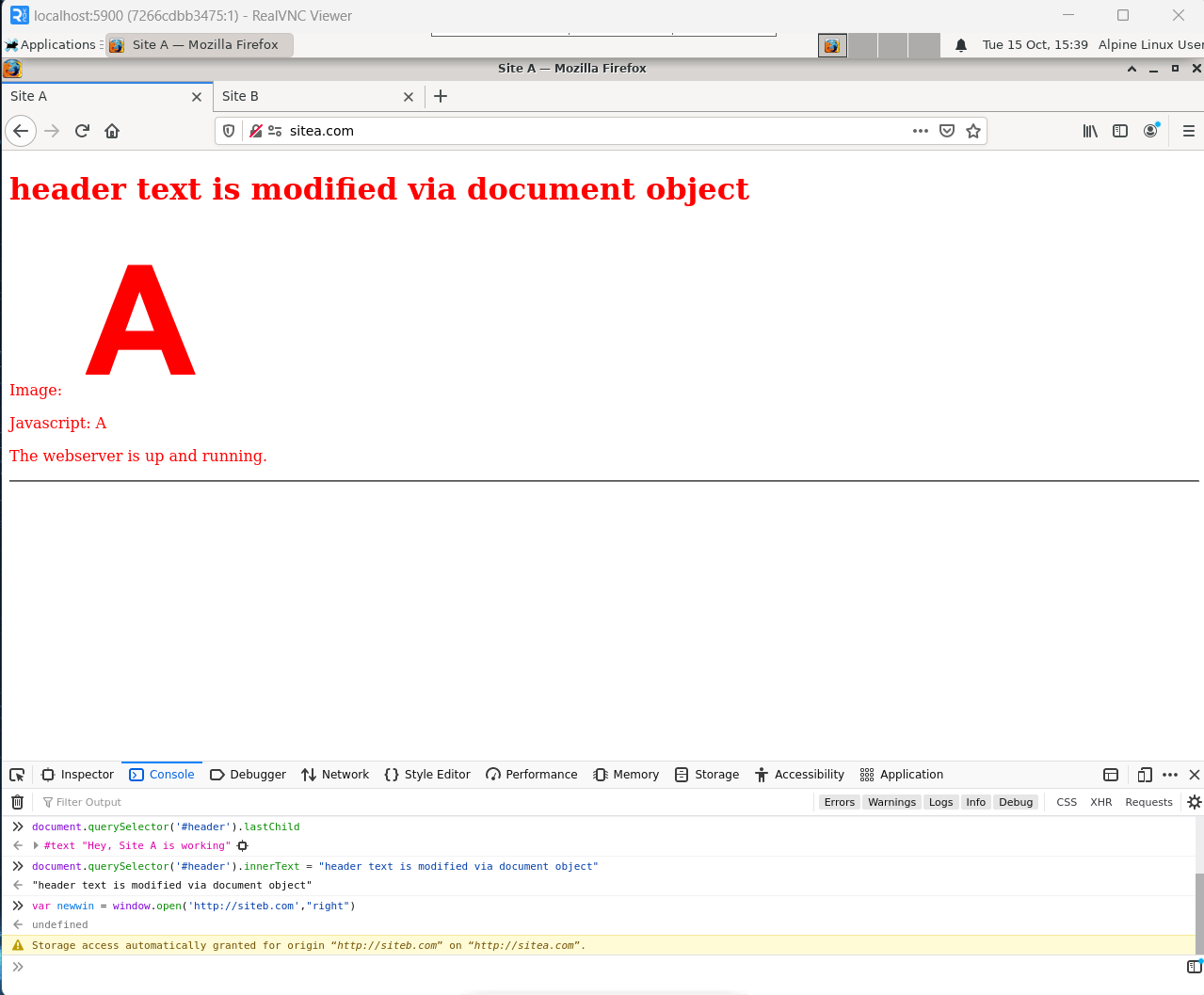
**b) Change the text “Hey,Site A is working” to “Header text is modified via document objeject”**



**c) Create a newwin variable for a new tab:**

**var newwin = window.open(http://siteb.com,”right”). Can you access document object of**

**newwin? Explain why if you could not**



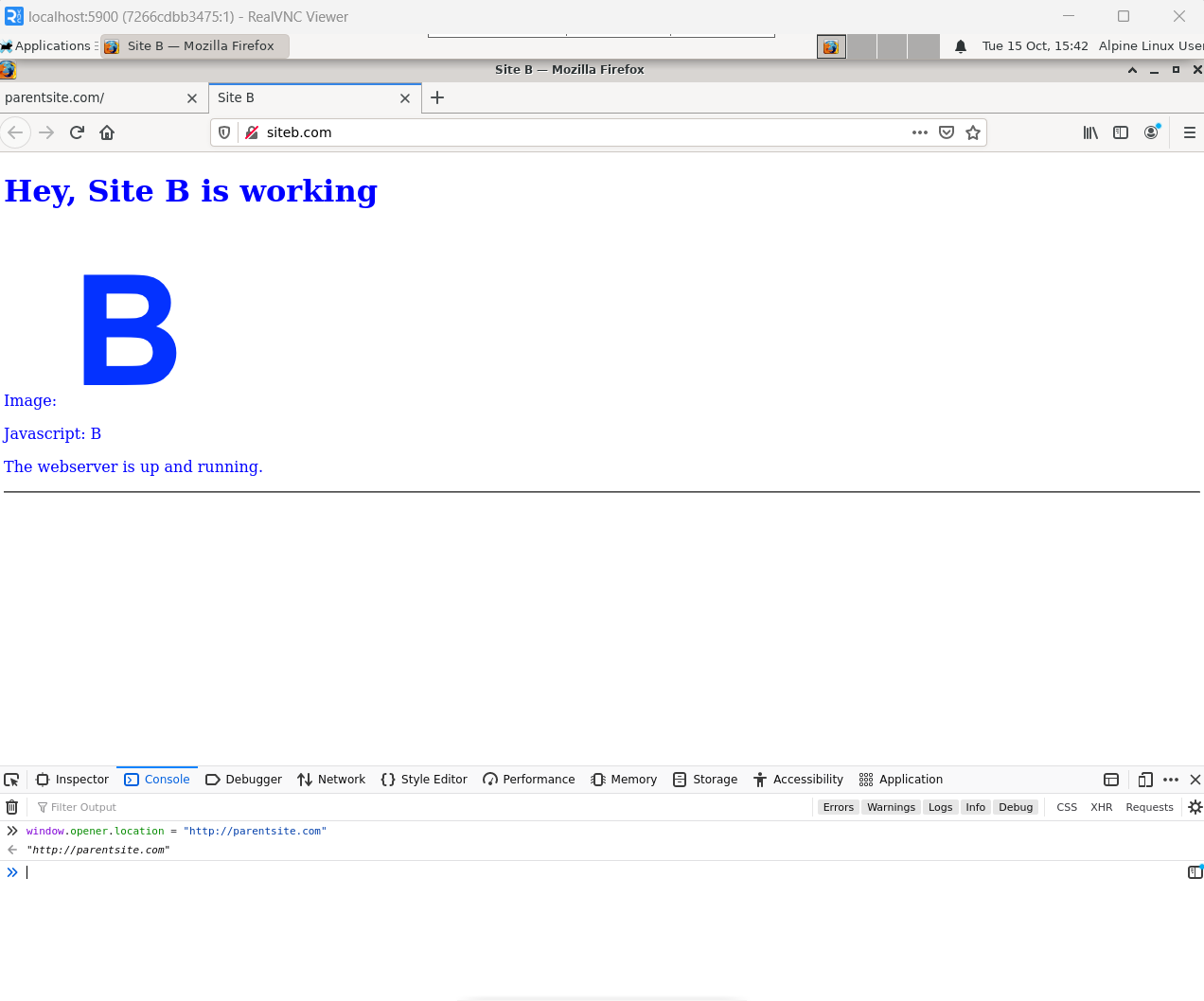
We cannot access document object of newwin



Because SOP (Same Origin Policy) prevents us from accessing object from cross-site.

**d) Enter JS code to set the new location for tab sitea which is the opener of current tab:**

window.opener.location=<http://parentsite.com>



**e) Try JS code to access document.body object of window.opener. Explain why if you could not**

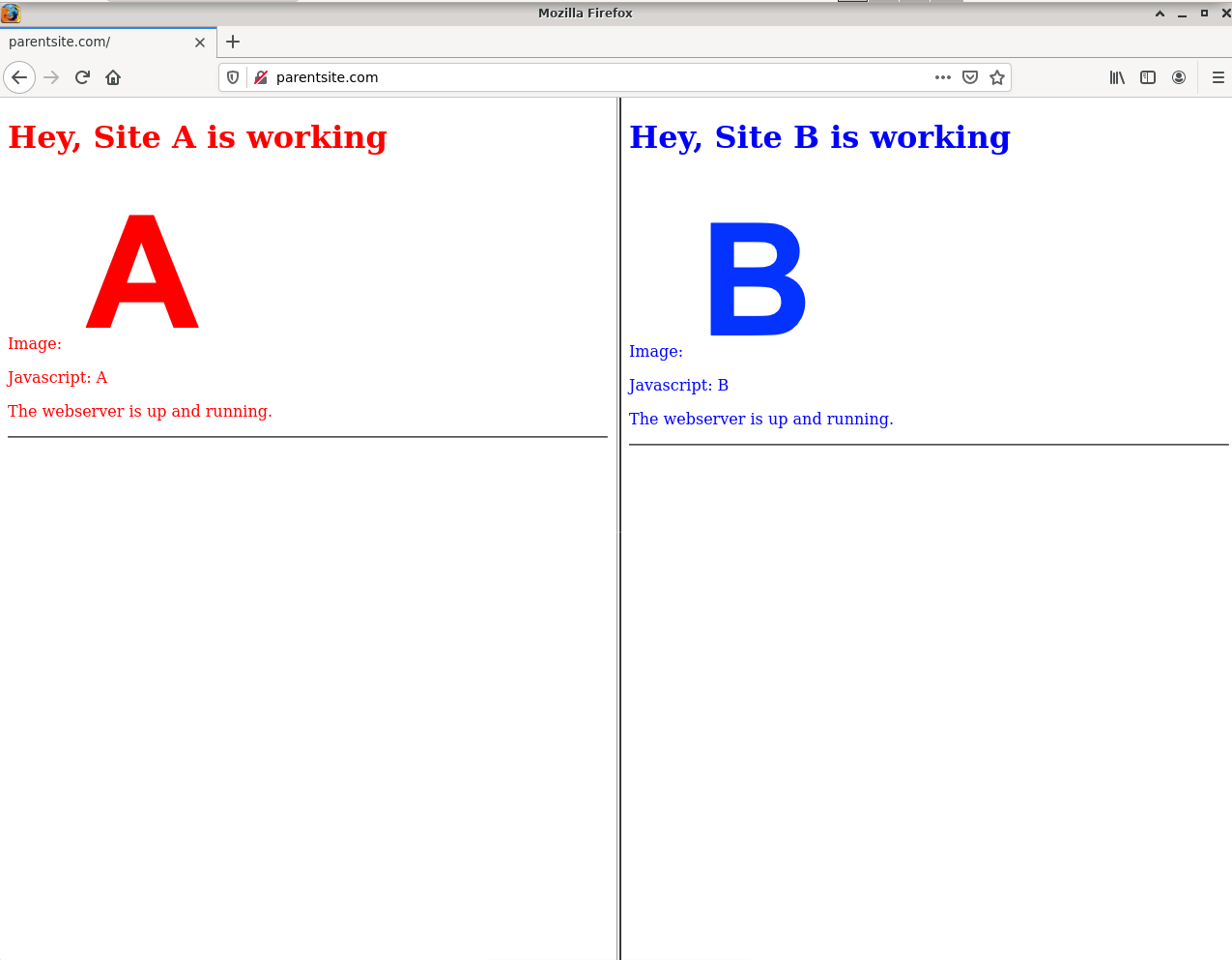


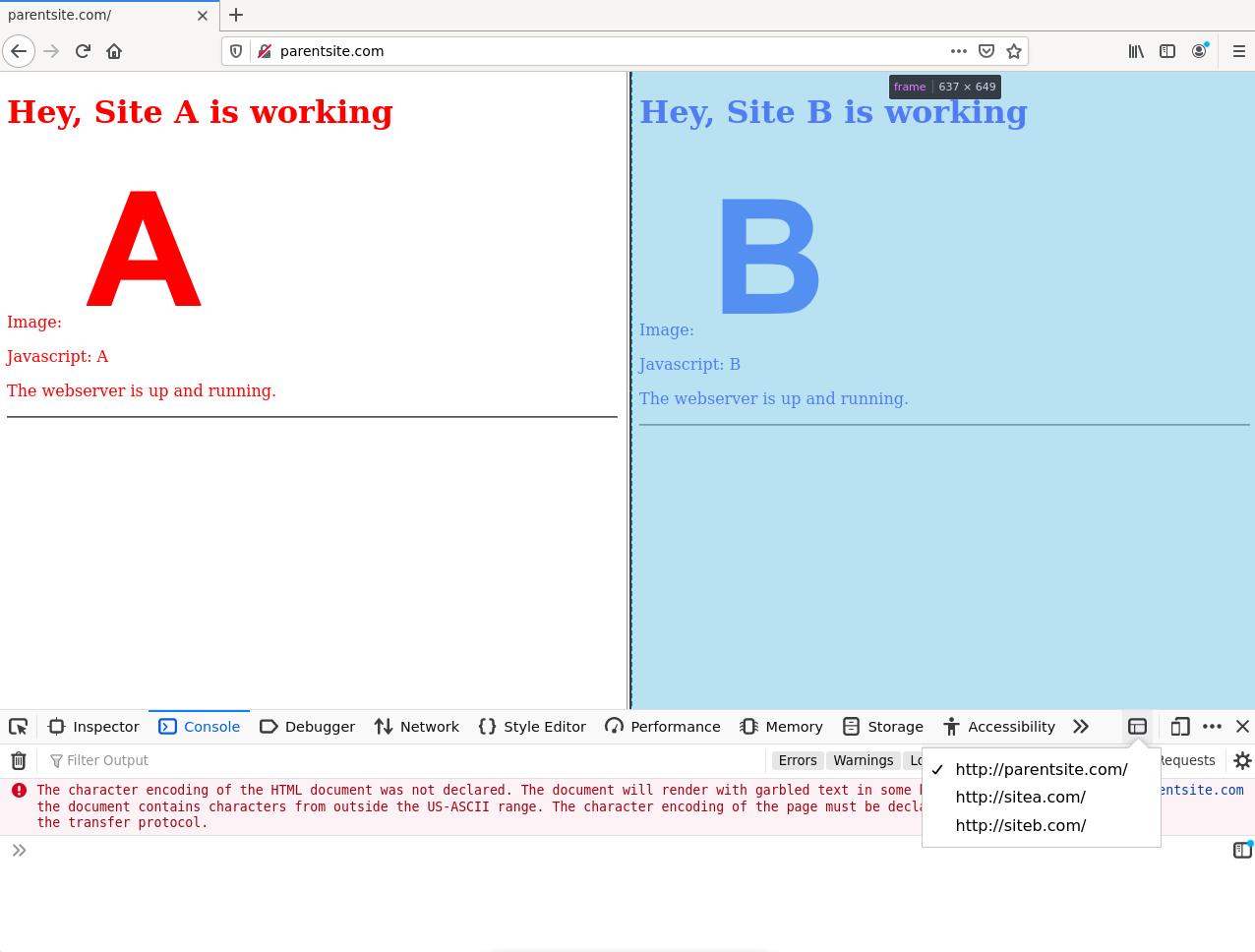
We cannot access document.body of window.opener. Because SOP prevents us from accessing object from other site not the same domain

**1.3 Frame Objects**

**Access http://parentsite.com then choose parentsite in list of sites on iframe**

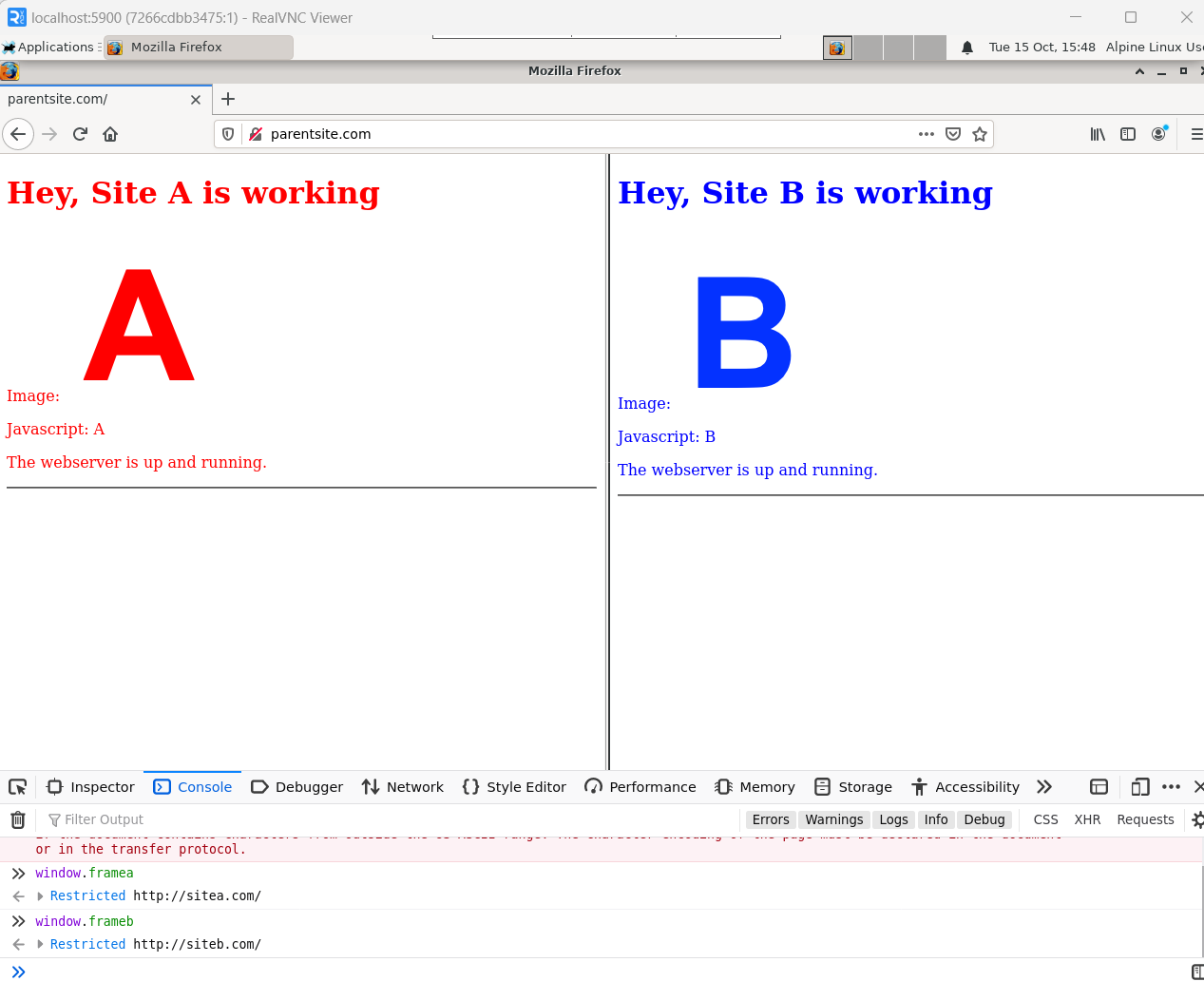
**picker.**





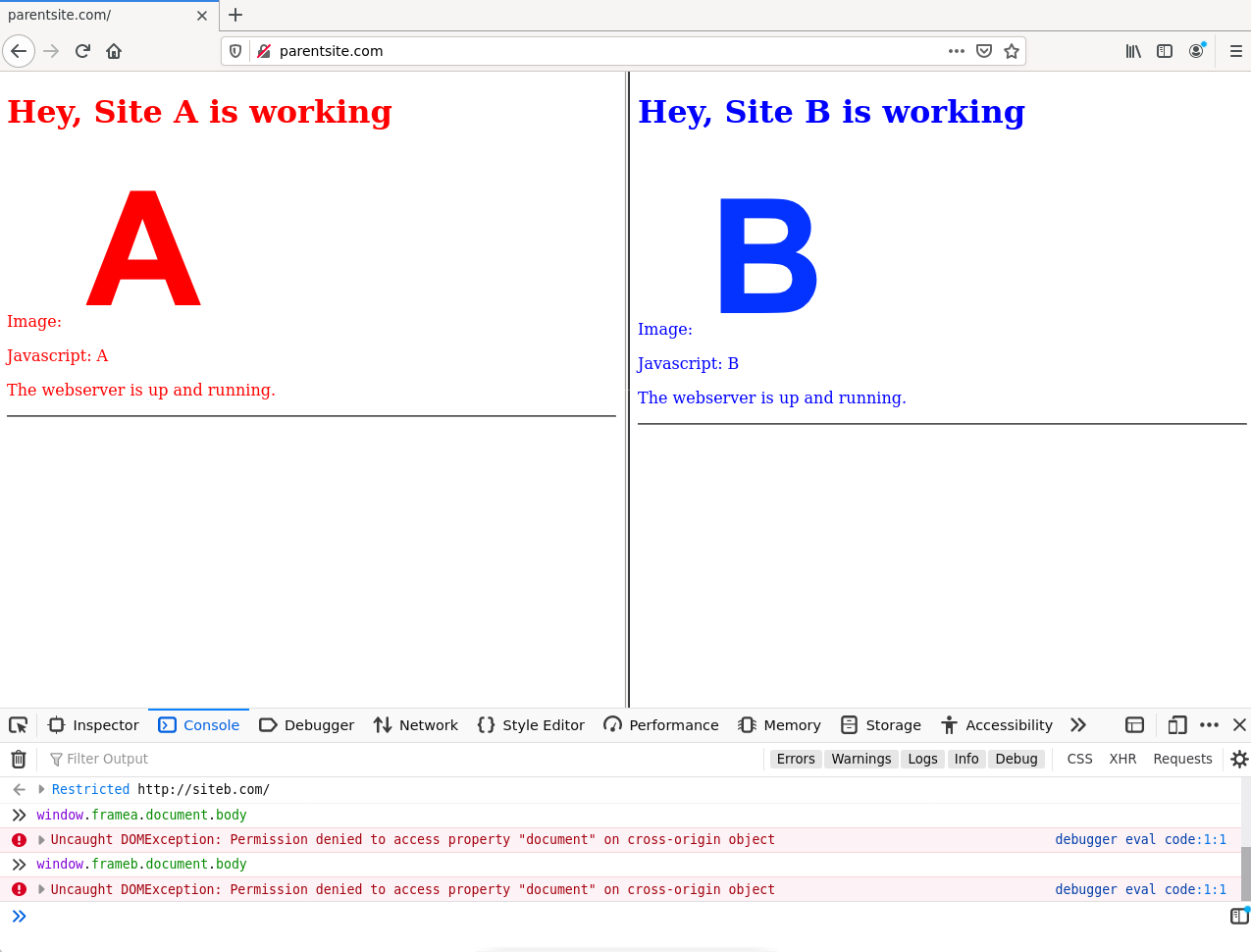
**Can you access window.framea, window.frameb?**

Yes, I can access window.framea and window.frameb

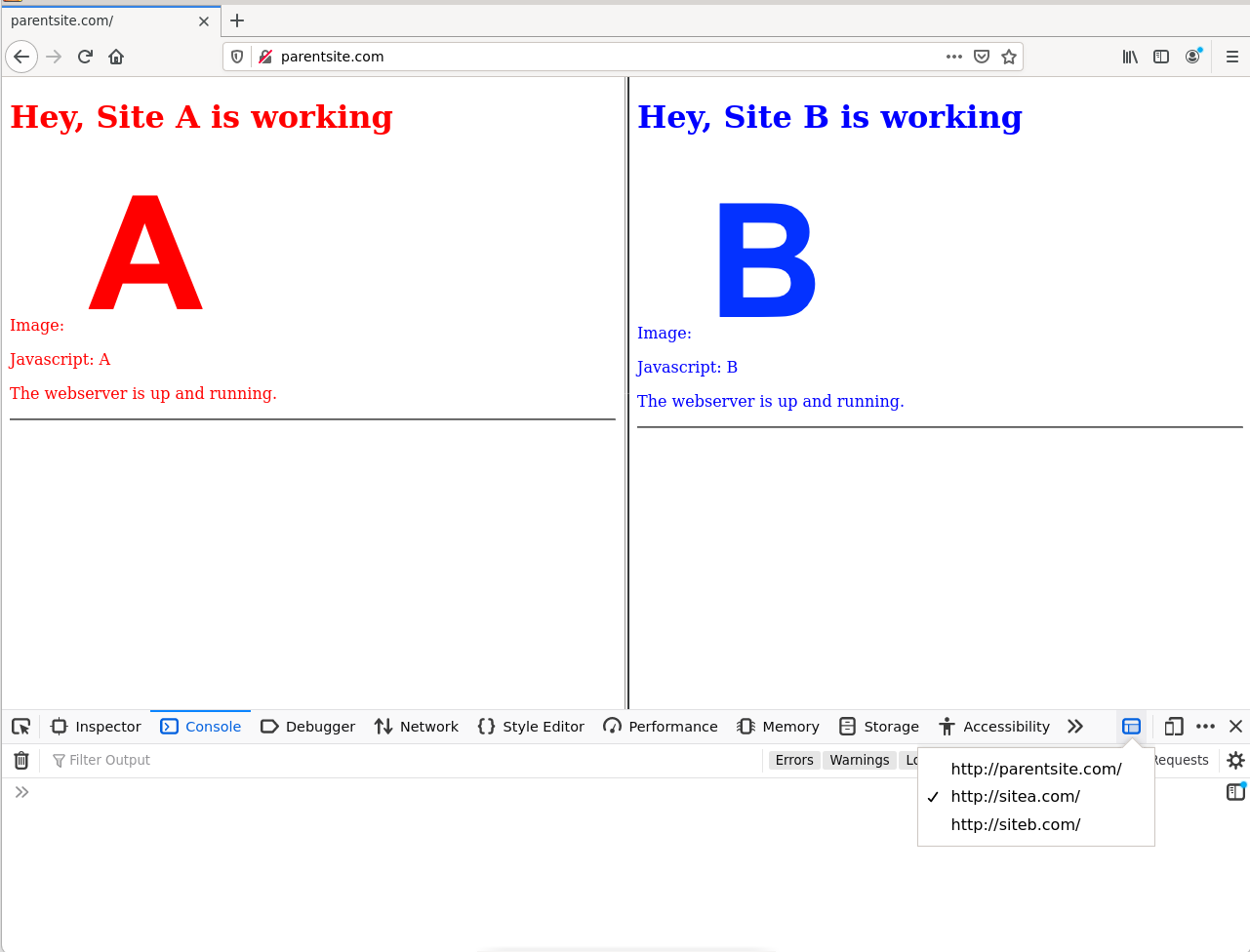


**b) Can you access document.body of framea, frameb from parentsite?**

No I cannot acces document.body of framea and frameb

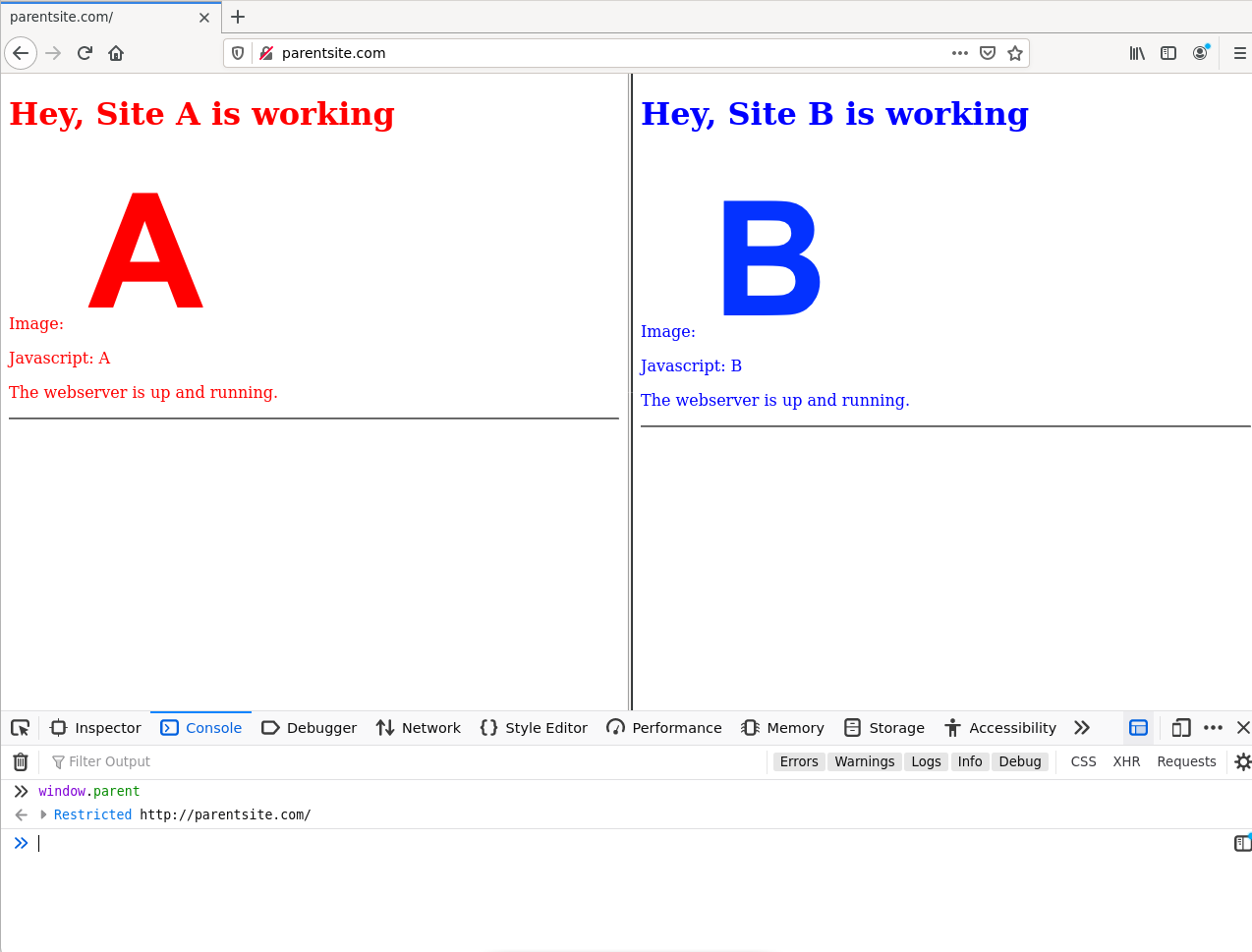


**- Select sitea.com in list of sites on iframe picker.**

****

**c) Can you access window.parentsite?**

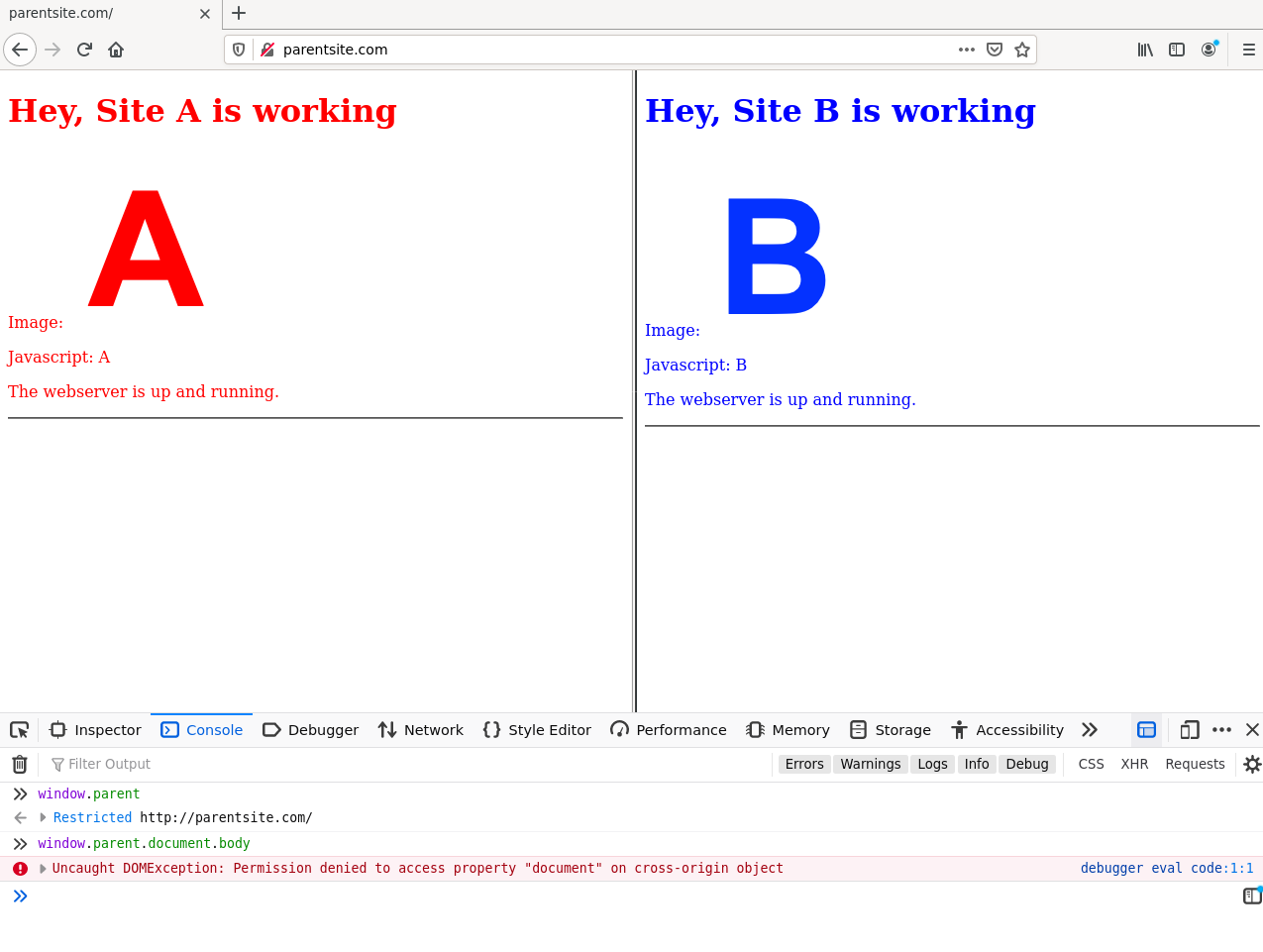
Yes I can access window.parent



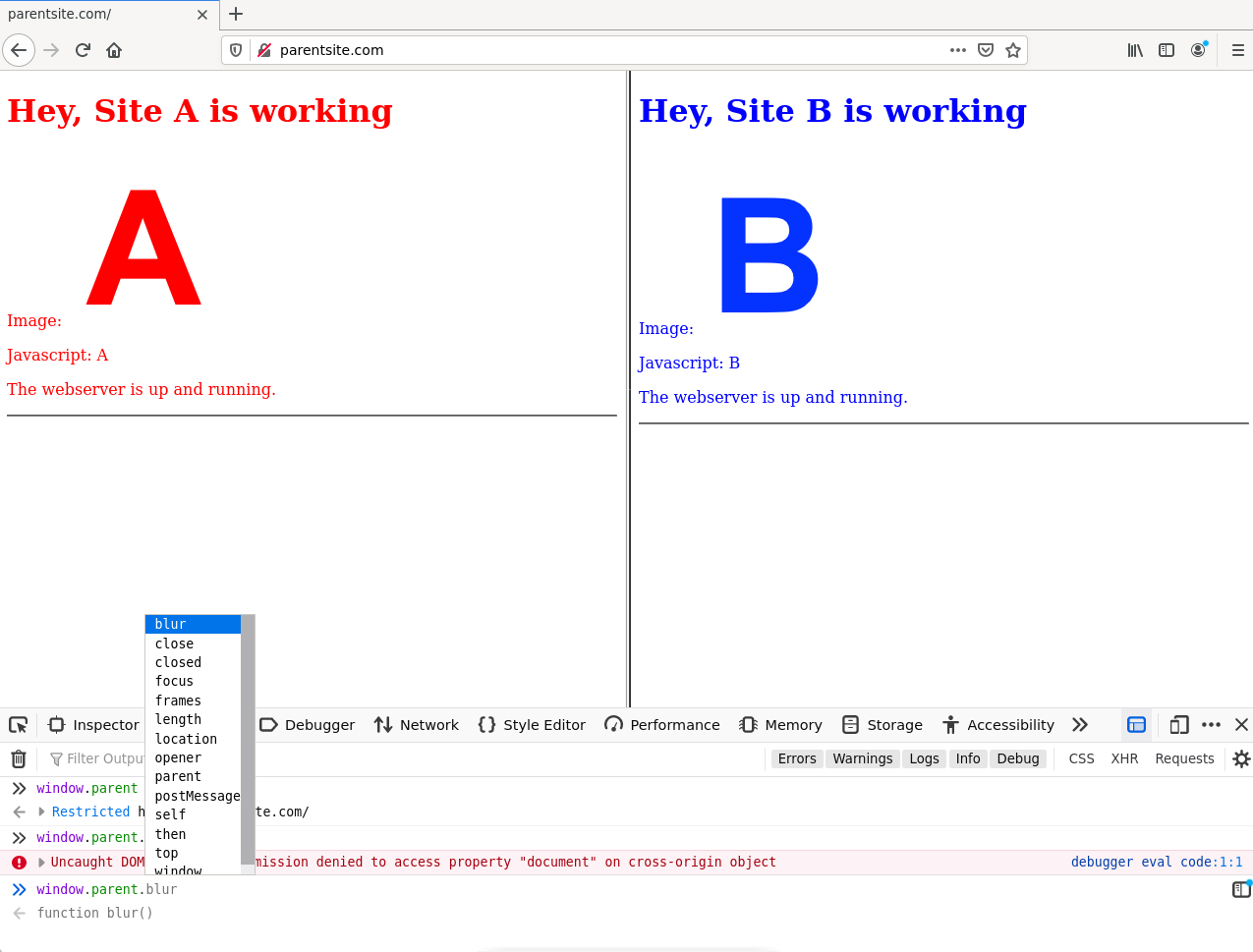
**d) Can you access document.body object of parentsite.com? Which object of parentsite that**

**sitea can access?**

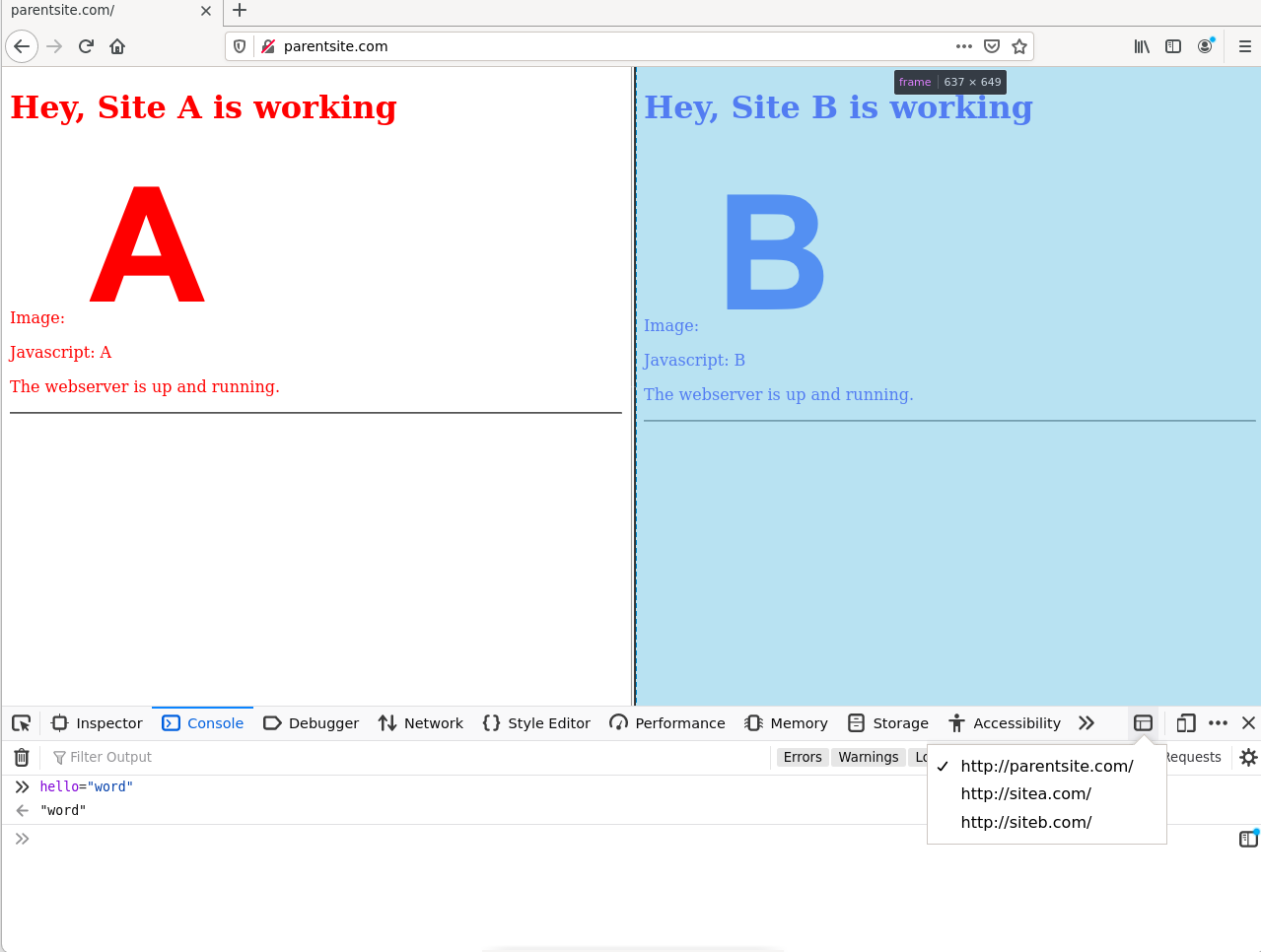
No I cannot access document.body object of parentsite.com



List of objects of parentsite I can access: blur, close, closed, focus, frames, length, location, opener, parent, postMessage, self, then, top, window

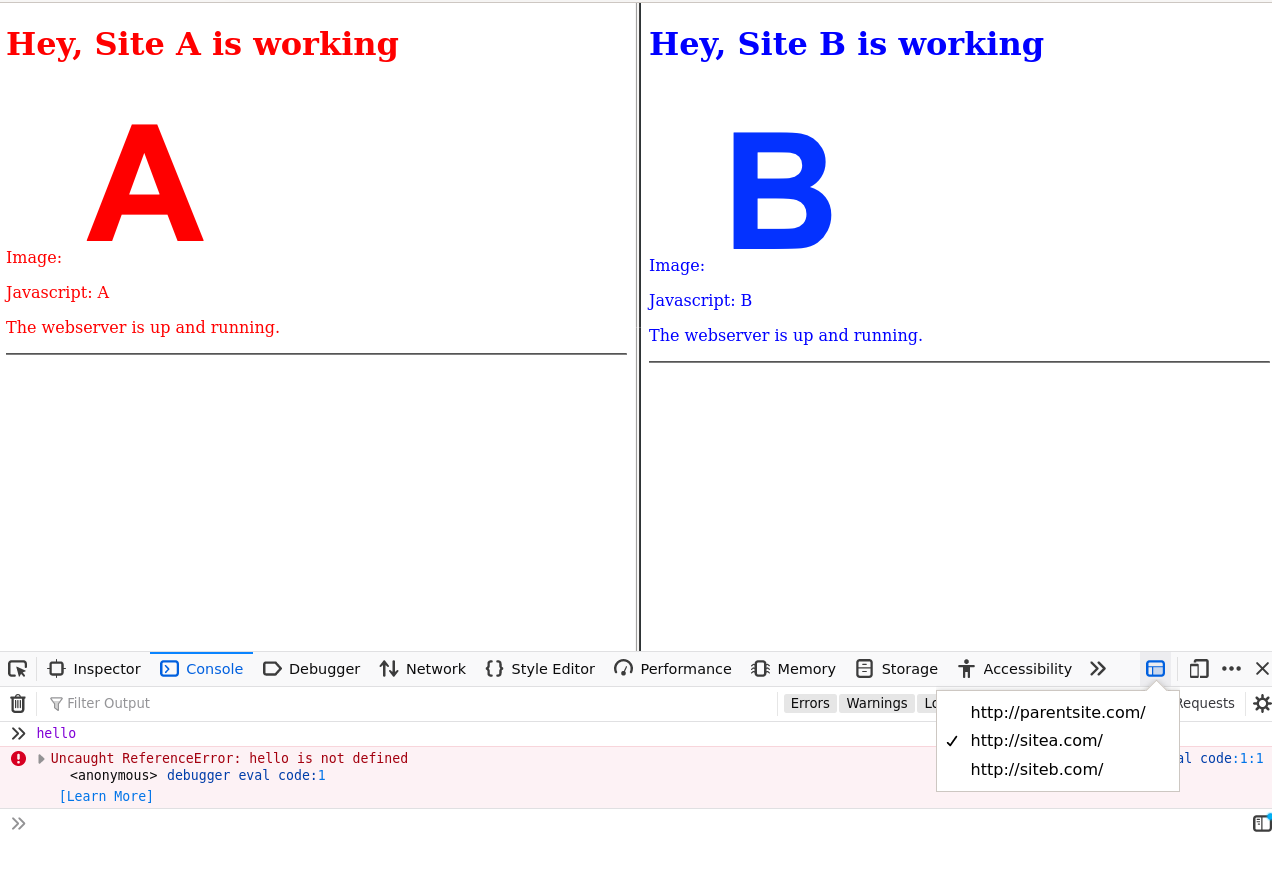


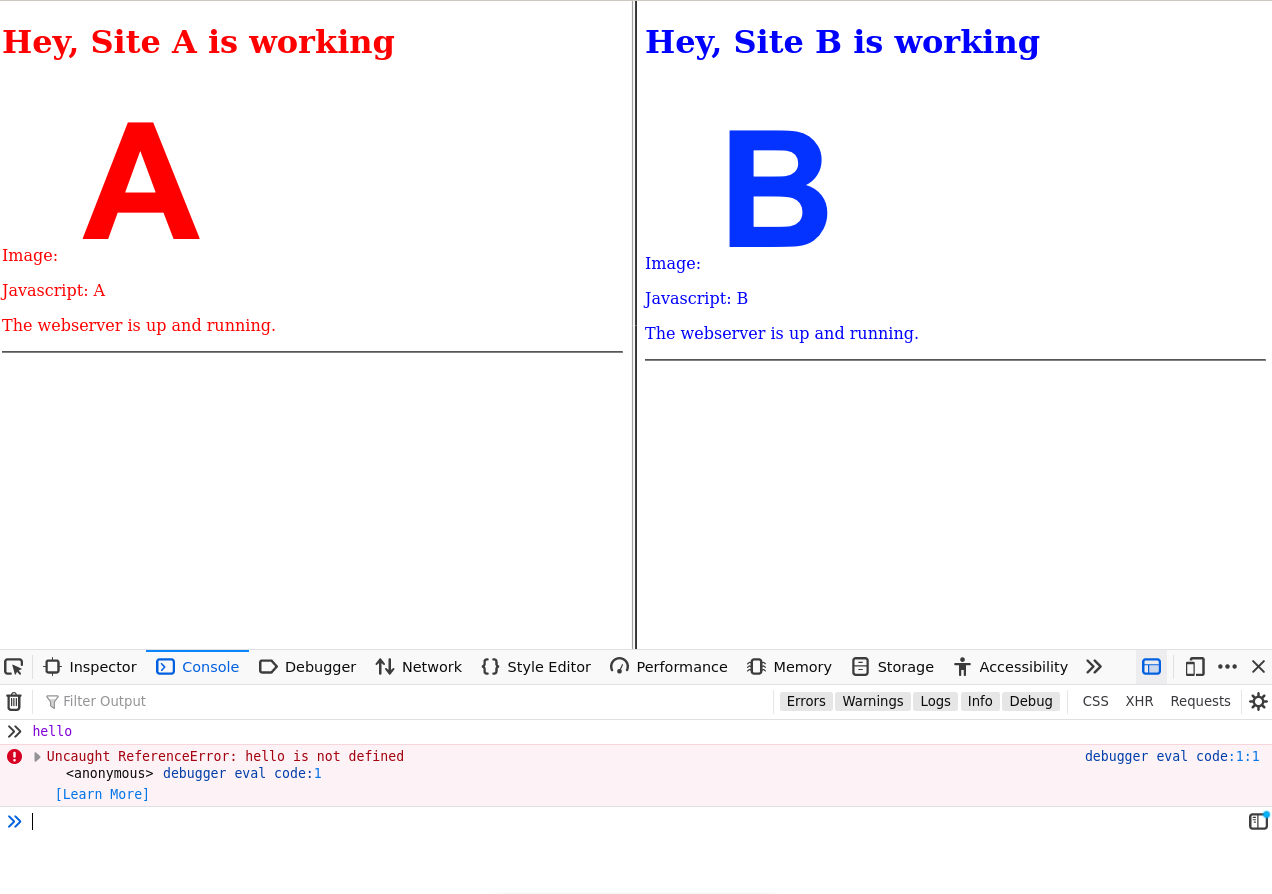
**- Select parentsite.com in list of sites on iframe picker, define a variable named hello: var hello=”world”**

****

**e) Can you access hello from sitea, siteb**

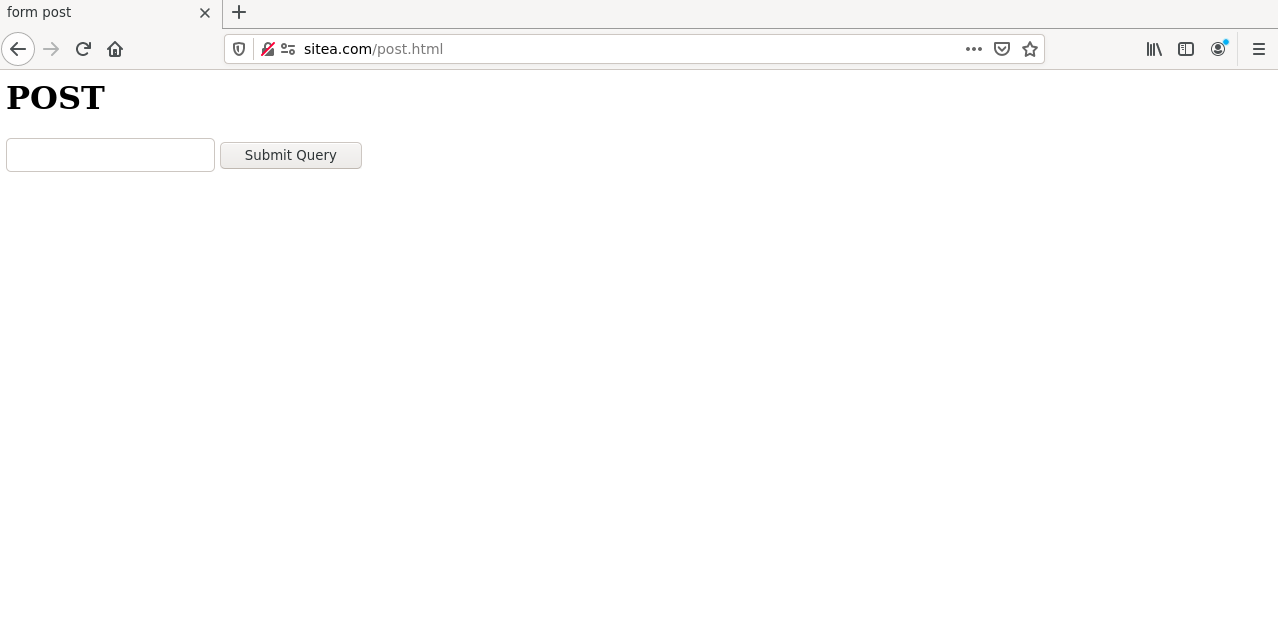
No,I cannot access hello from sitea and siteb





**1.4 Sending POST request to a site:**

**a)Access** [**http://sitea.com/post.html**](http://sitea.com/post.html)



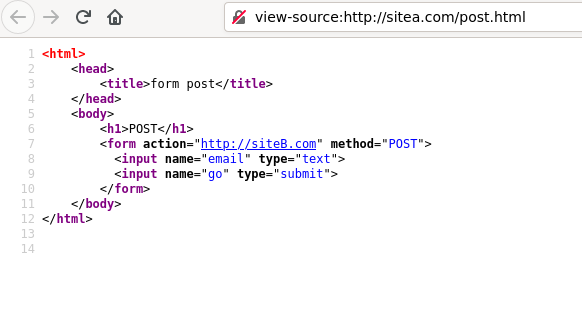
**b) Type some text in textbox then click**

**“Submit query”. What would show up?**

**Explain why you see that.**

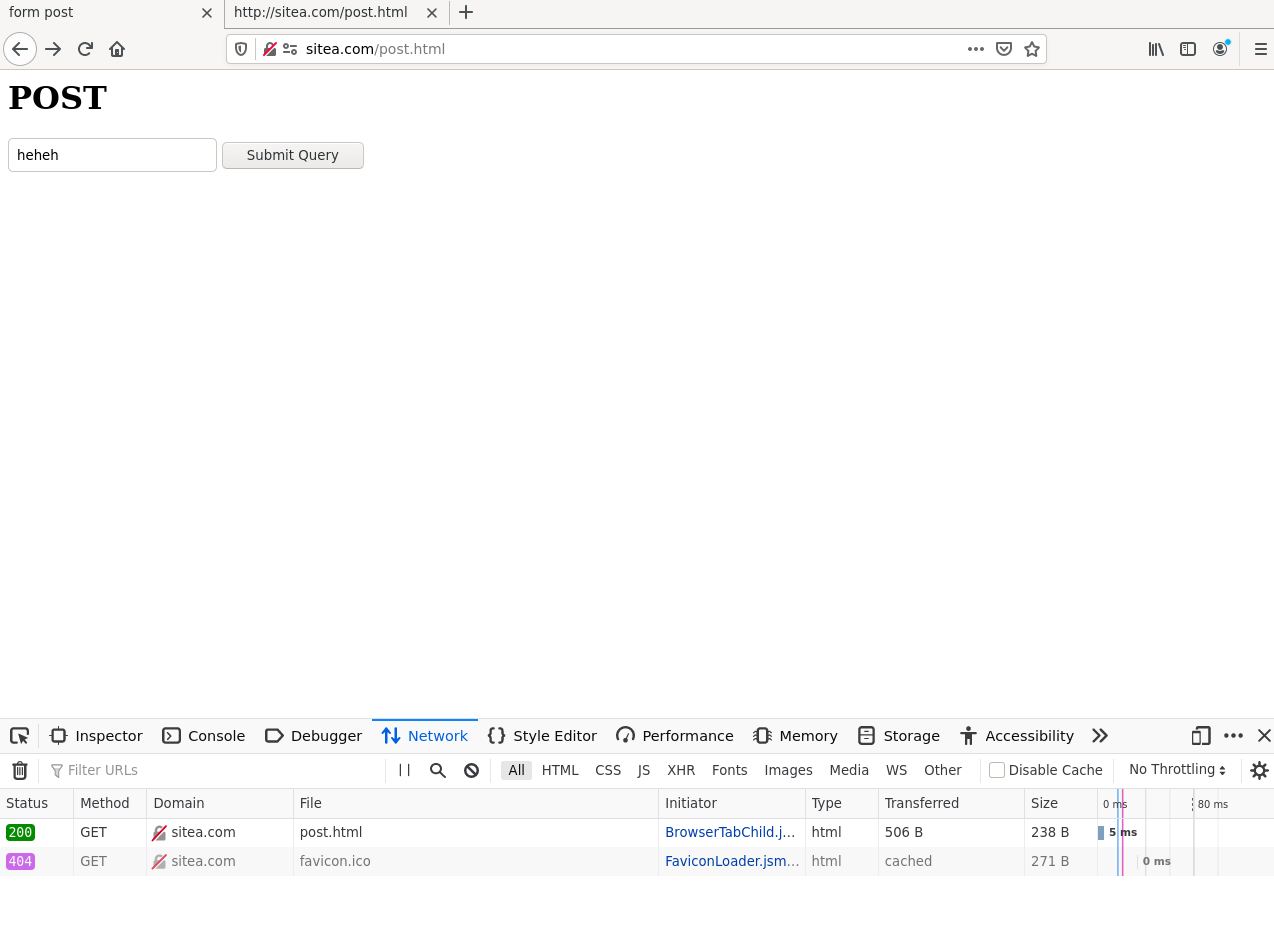
When I click “Submit query”, it will show siteb.com

Because as you can see in the page source of sitea post



It uses a form having action to htt://siteB.com. Because of that, when we submit, the form will navigate us to siteb.com!

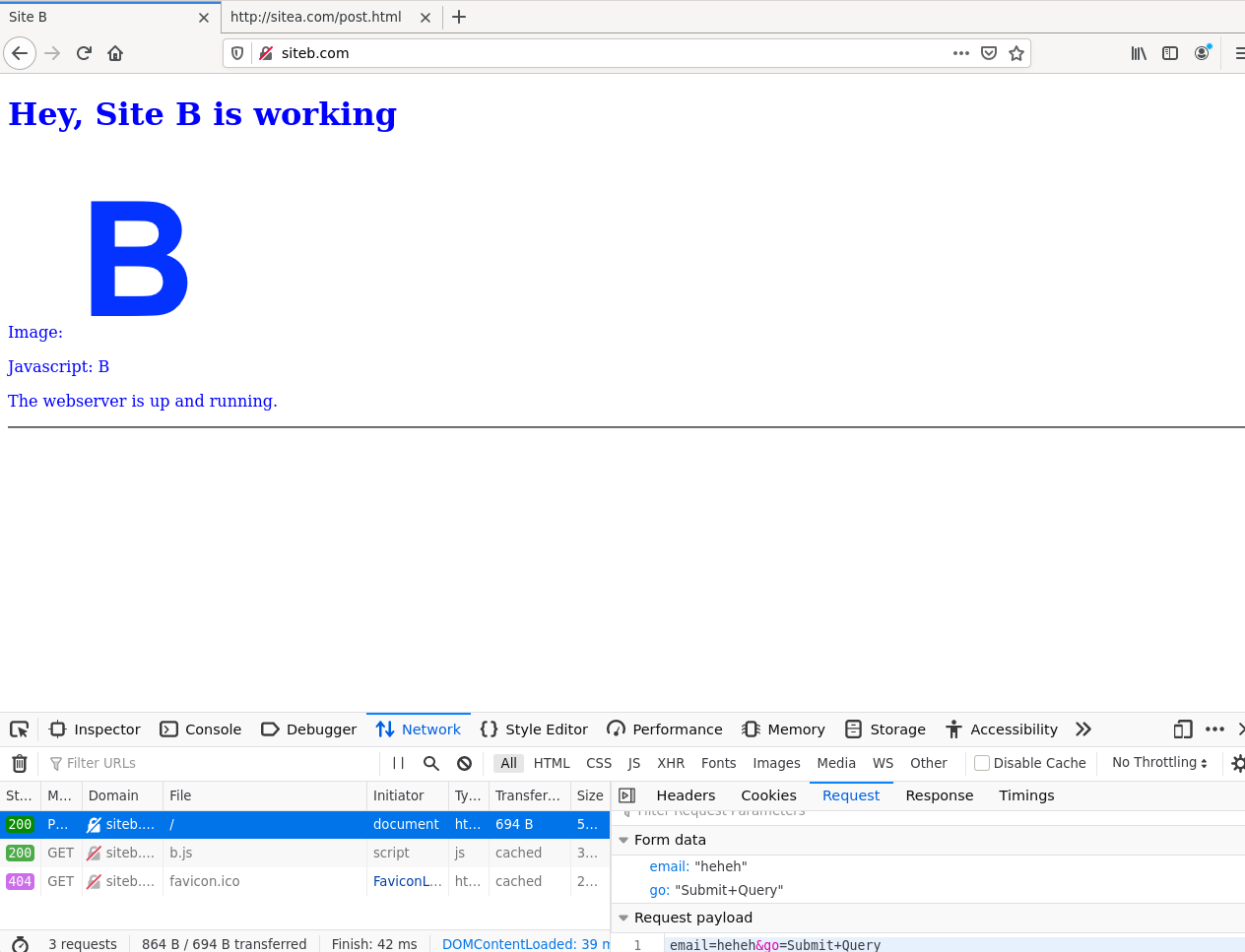
**c) Repeat step (a), toggle Web console >> Network (Ctrl + Shift + E)**

****

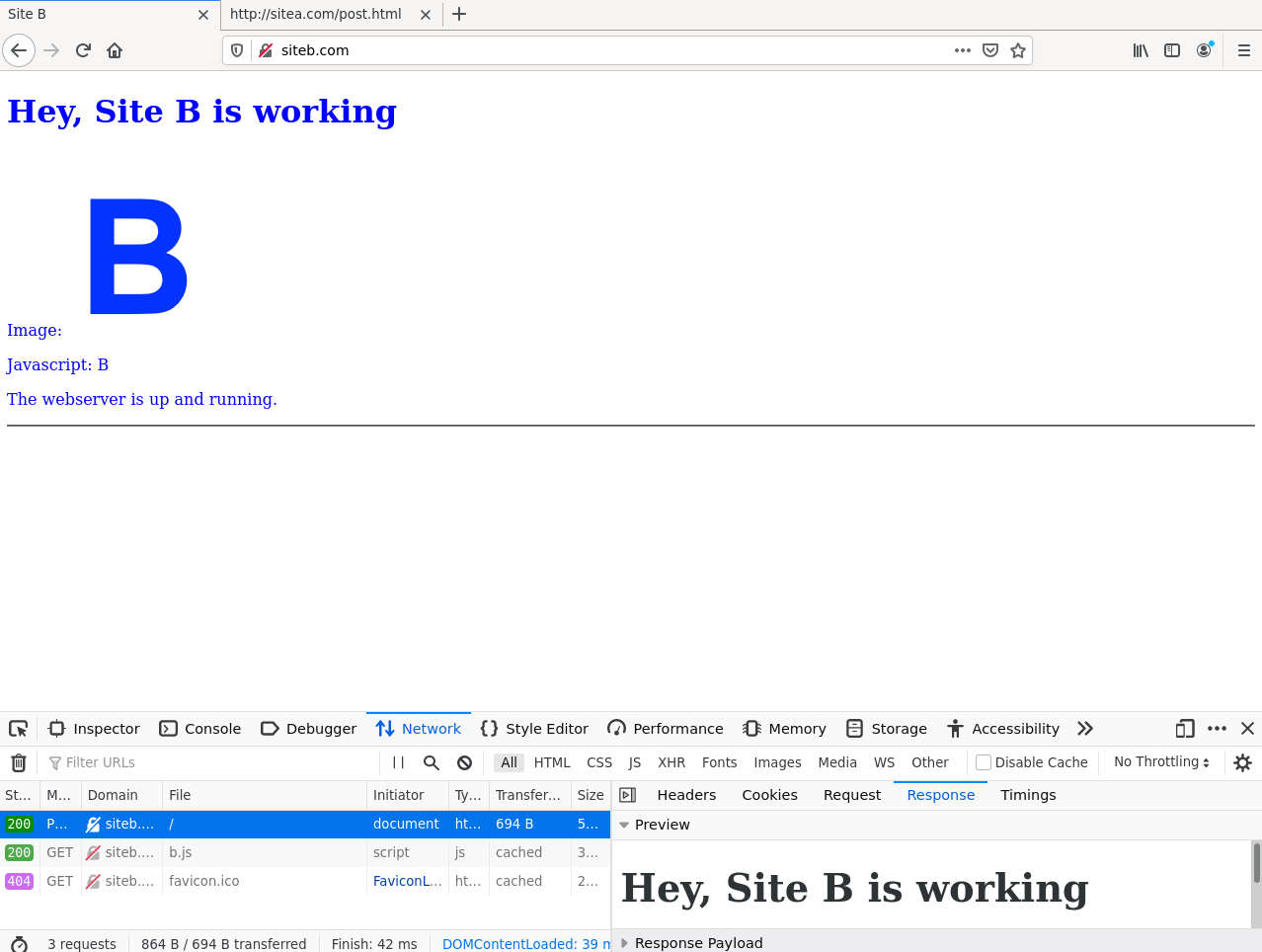
**d) Repeat step (b), examine the request and response header. What can you conclude about**

**POST form request.**

Request header:



Response Header:



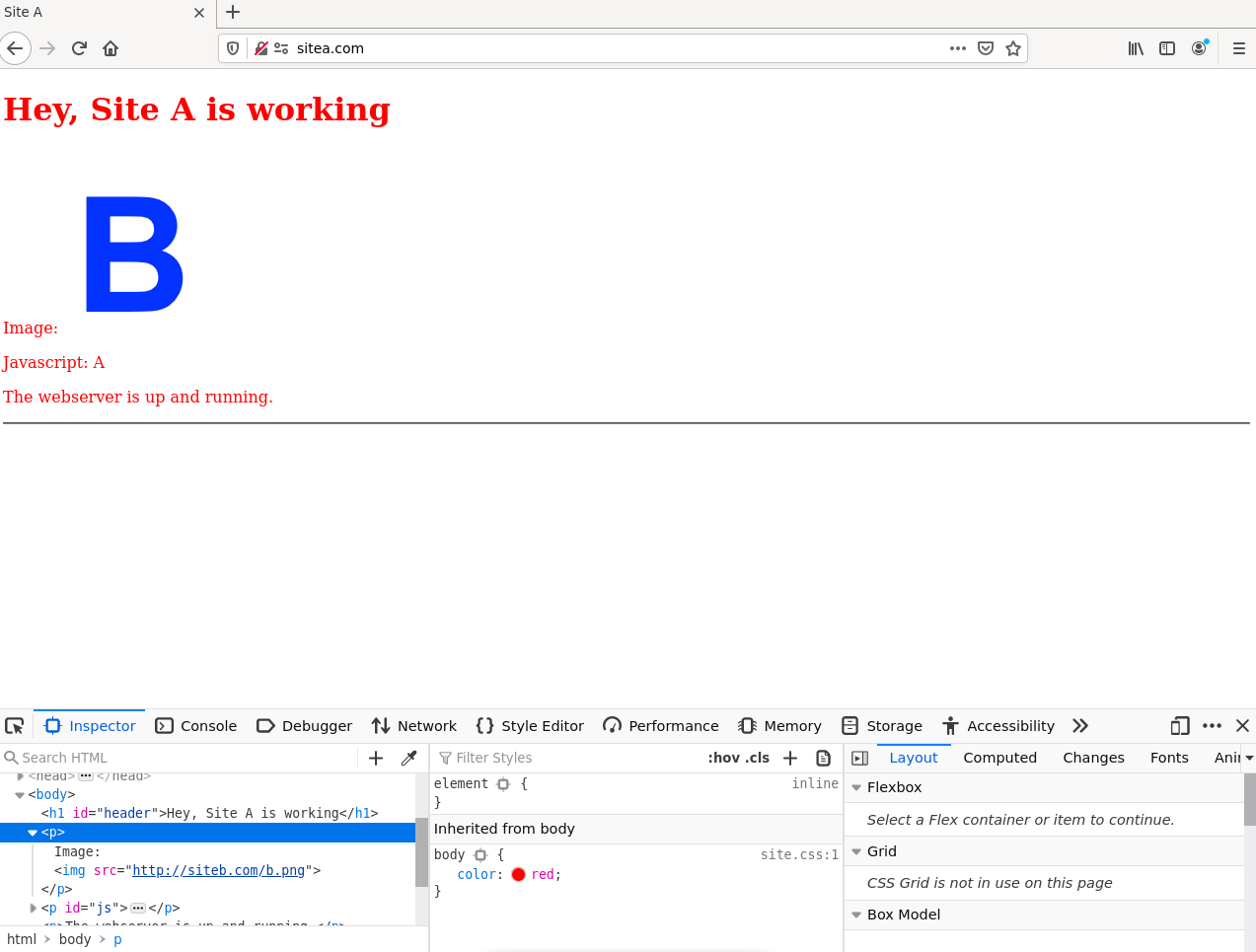
Conclusion about POST request:

-When we send data using POST request, the data will be contained in request.body!

**1.5 Access image,stylesheet from othersites**

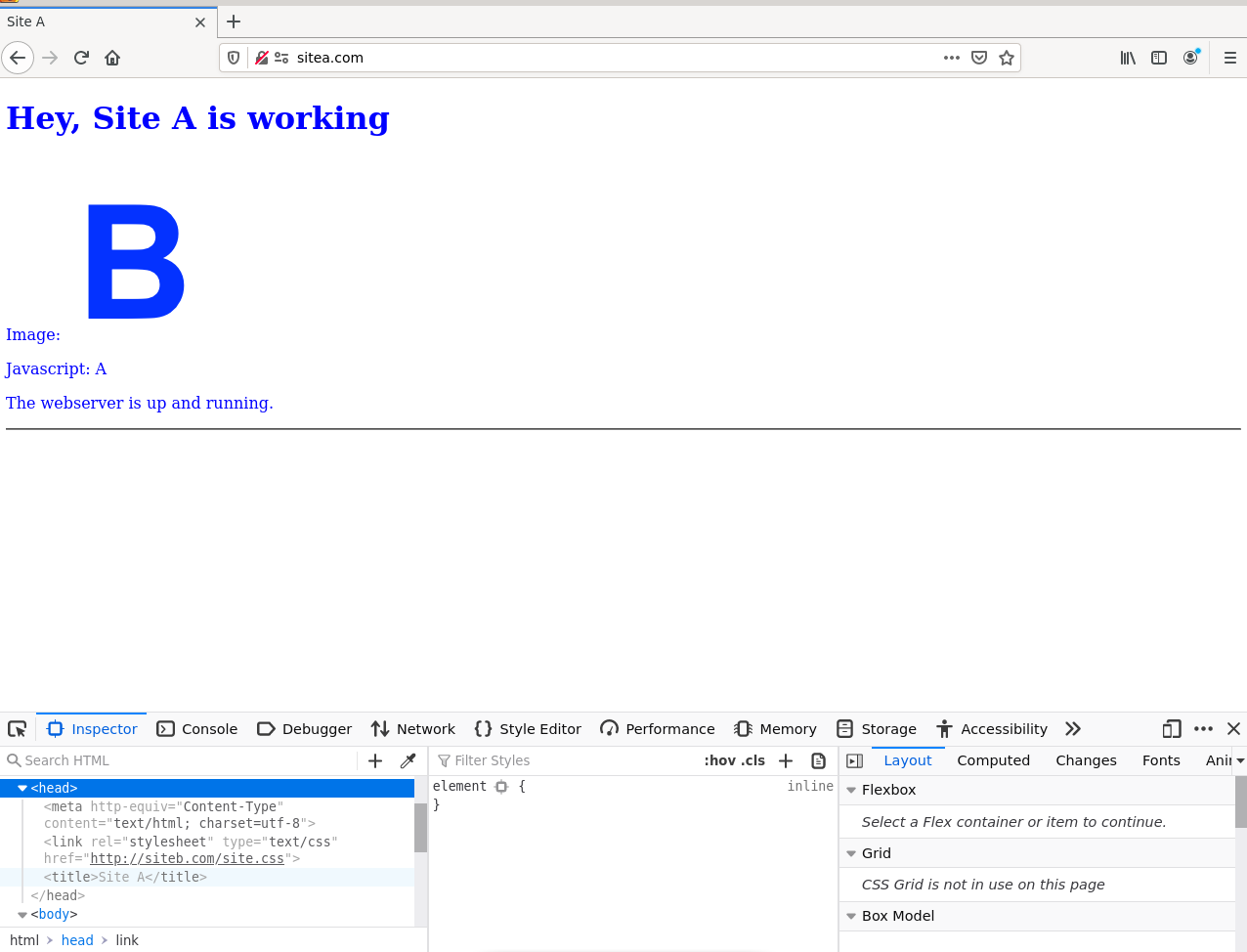
**a) In the Inspector panel of sitea.com, can you change sitea’s image to image from siteb.com?**

Yes, I can change sitea’s image to siteb.com by modify source of img in sitea to be <http://siteb.com/b.png>



**b) In the Inspector panel of sitea.com, can you change the stylesheet to that of siteb.com**

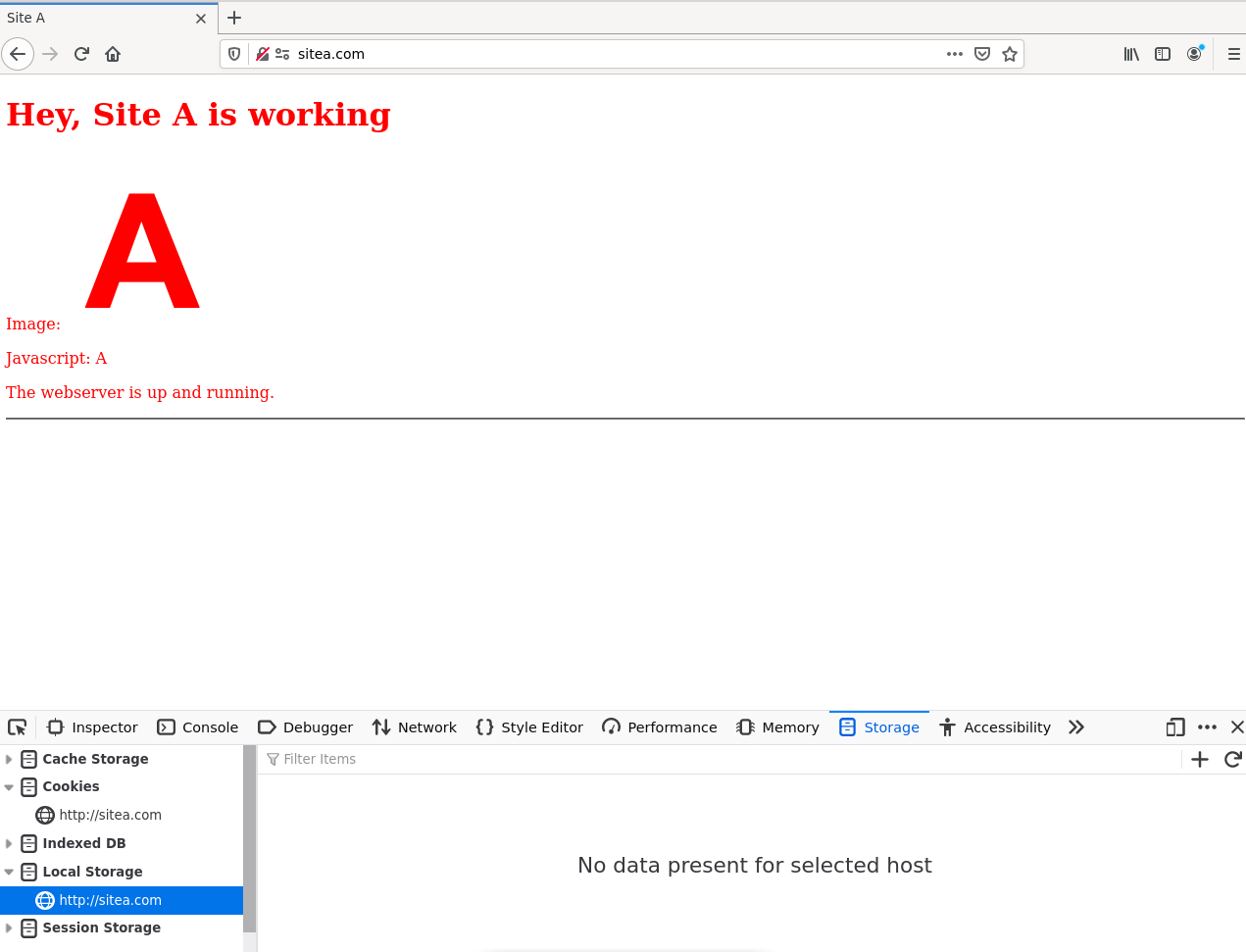
Yes, I can change sitea’s stylesheet to siteb’s stylesheet by modify hef to http://siteb.com/site.css



**1.6 SOP applies to web storage**

**1.6.1 Local Storage**

**Load www.sitea.com in one tab, open Web developer panel > Storage, check that the LocalStorage is empty (Storage >> Local Storage).**

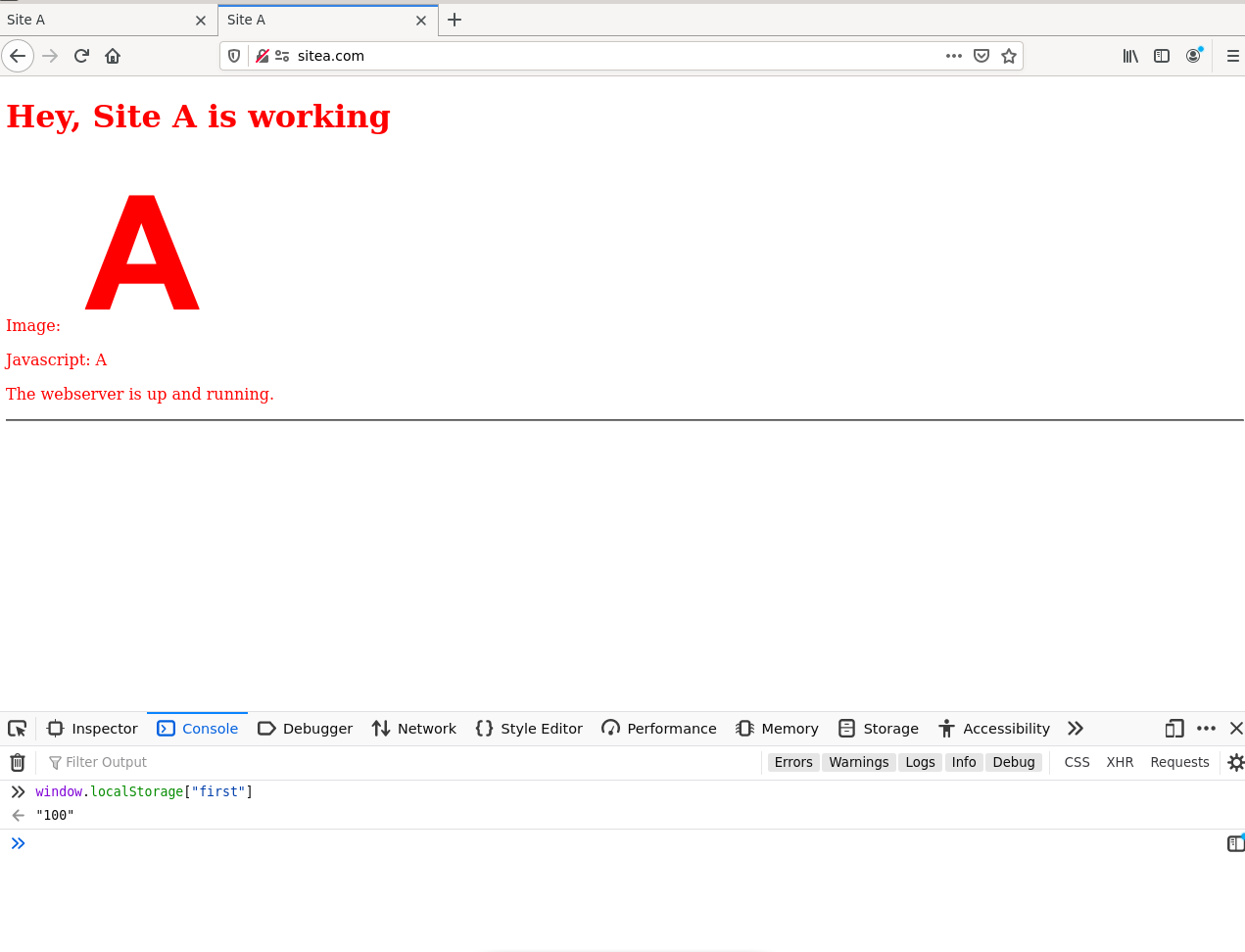
****

**a) Define a new variable named first = 10. You can define either: with javascript in console: window.LocalStorage[“first”] = 100 or Add Item in Storage tab**

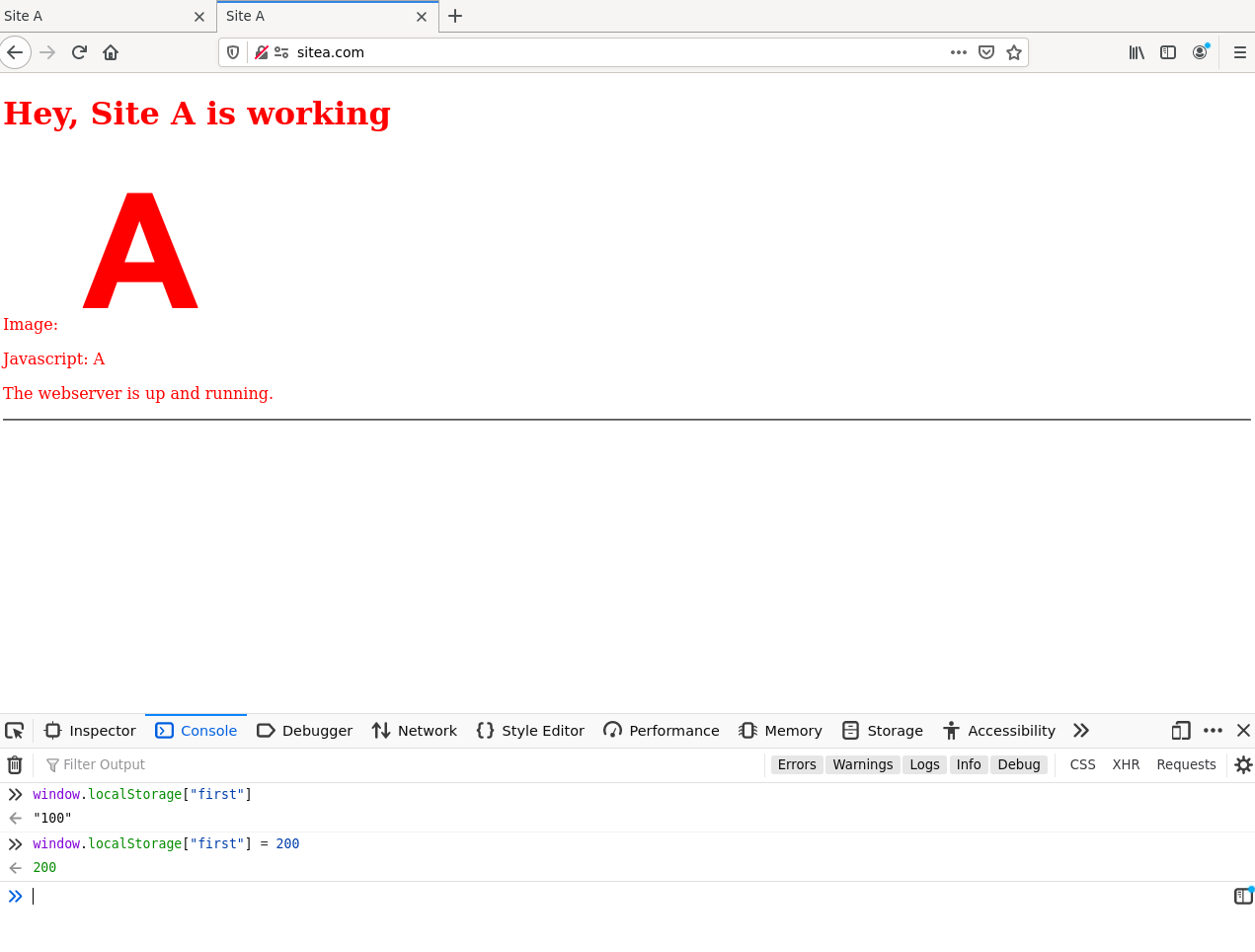
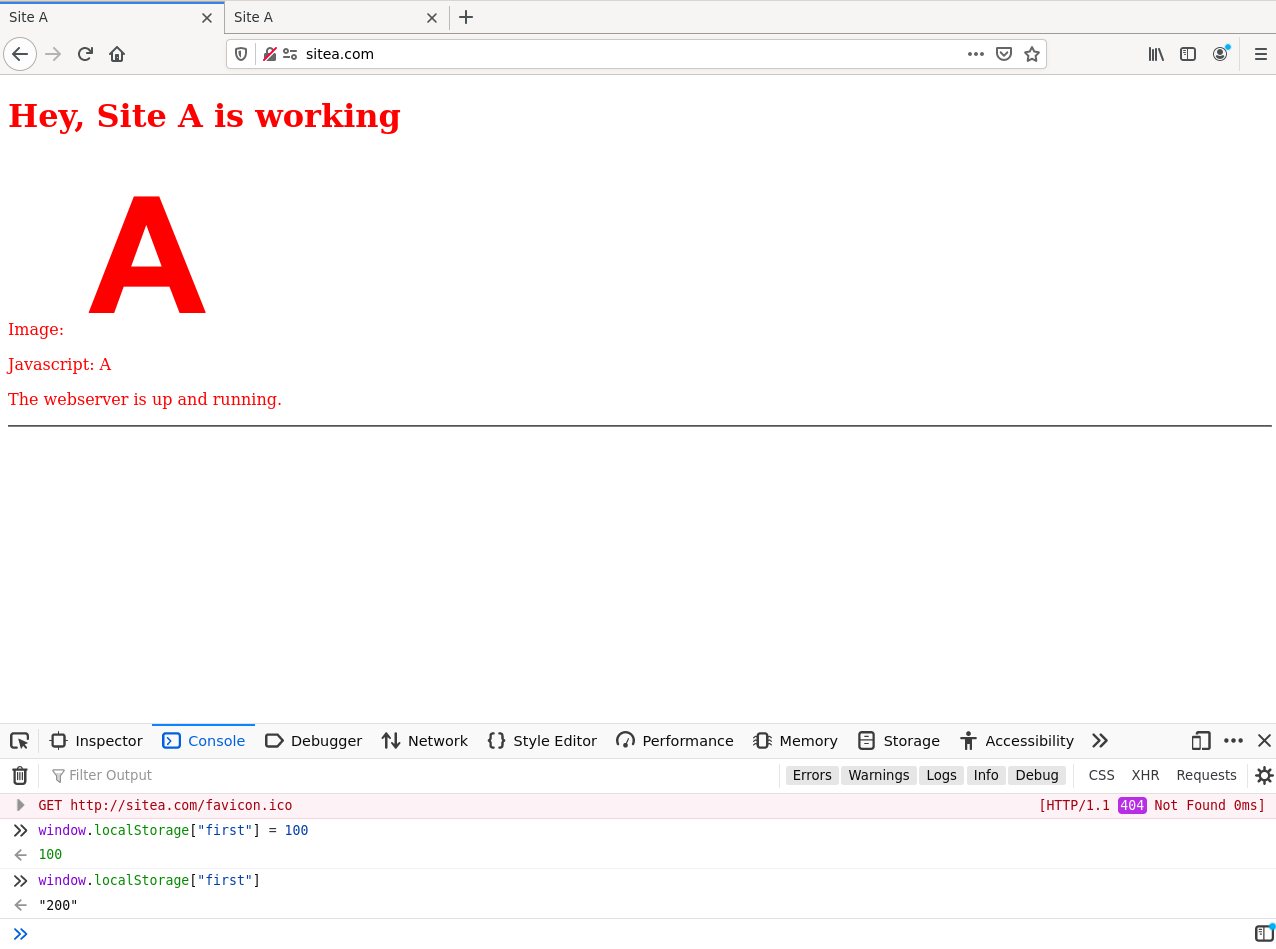
****

**b) Open sitea.com in a new tab. Can you access first variable in this tab? Change first to a new value, go back to the previous sitea.com tab, what is the value of first then.**

Yes I can access first variable in new tab



I changed first to 200

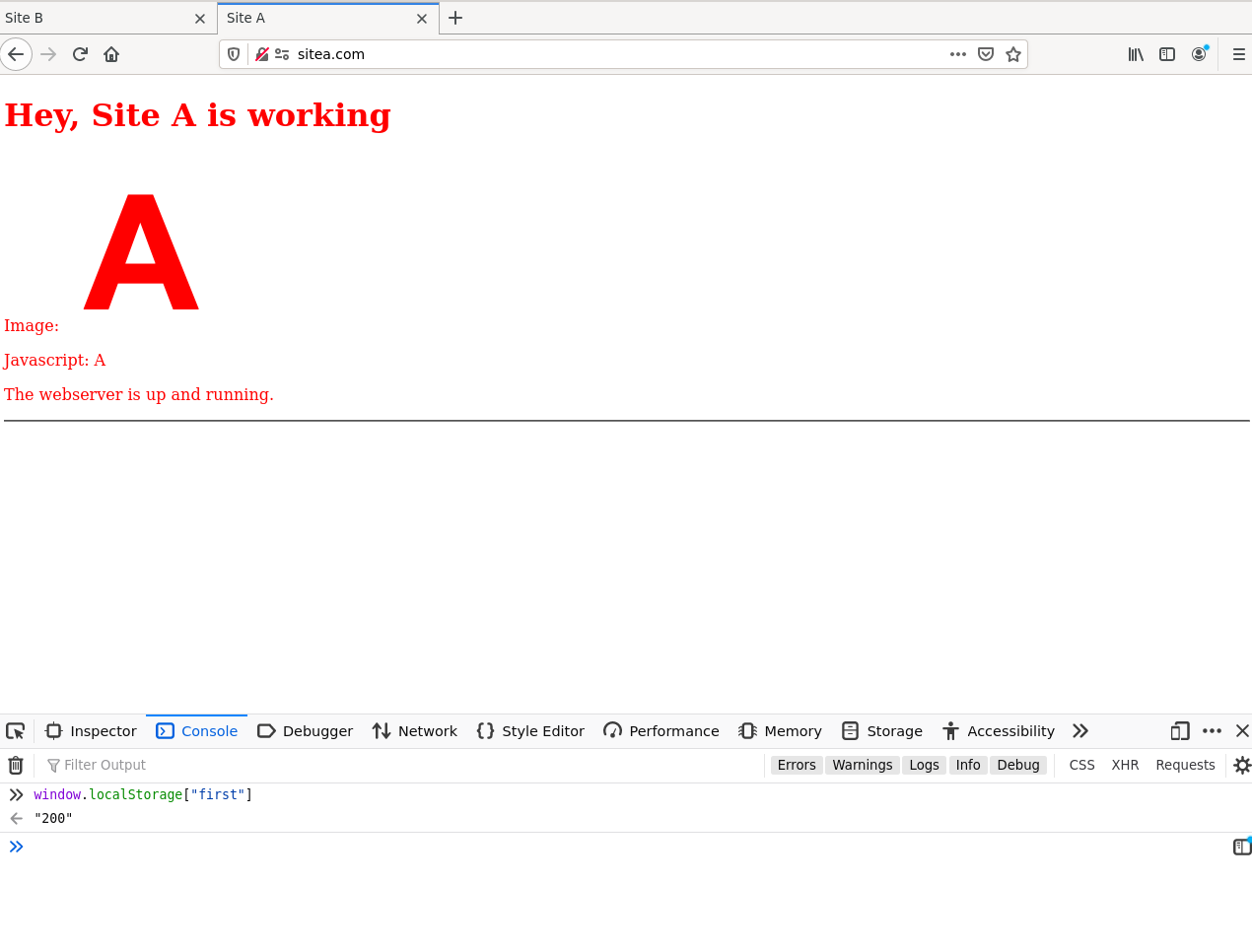
* 
* First in previous tab
* 
* The value has been changed!

**c) Can you access first from siteb.com?**

No, I cant access first from siteb.com

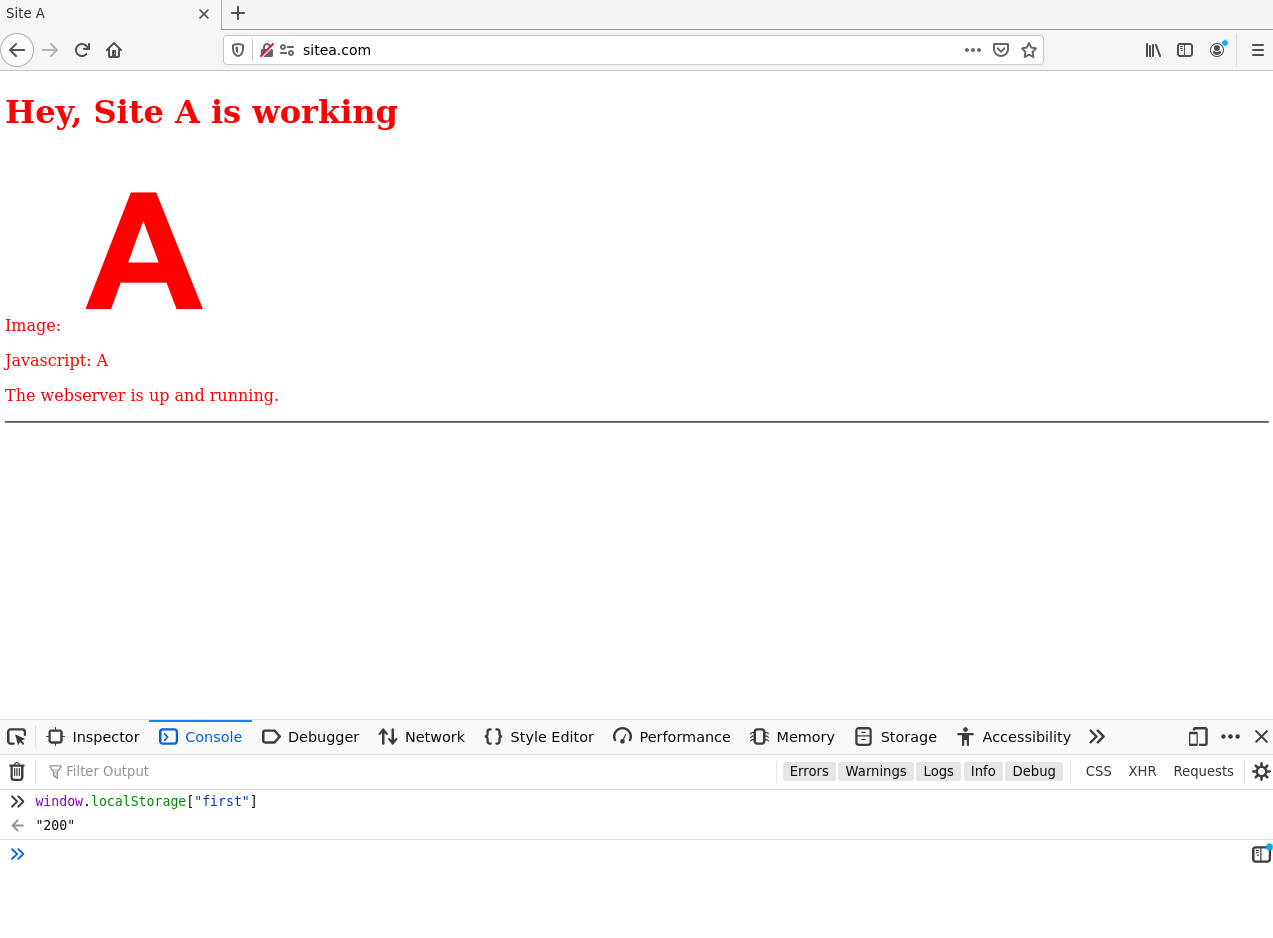
****

**d) Close all sitea.com tabs. How about first variable when sitea.com is re-opened.**

****

The first variable is still there in localStorage

**e) Close browser then re-open, does first variable still exist?**

****

Yes the variable still exists

**f) What can you conclude about Local Storage?**

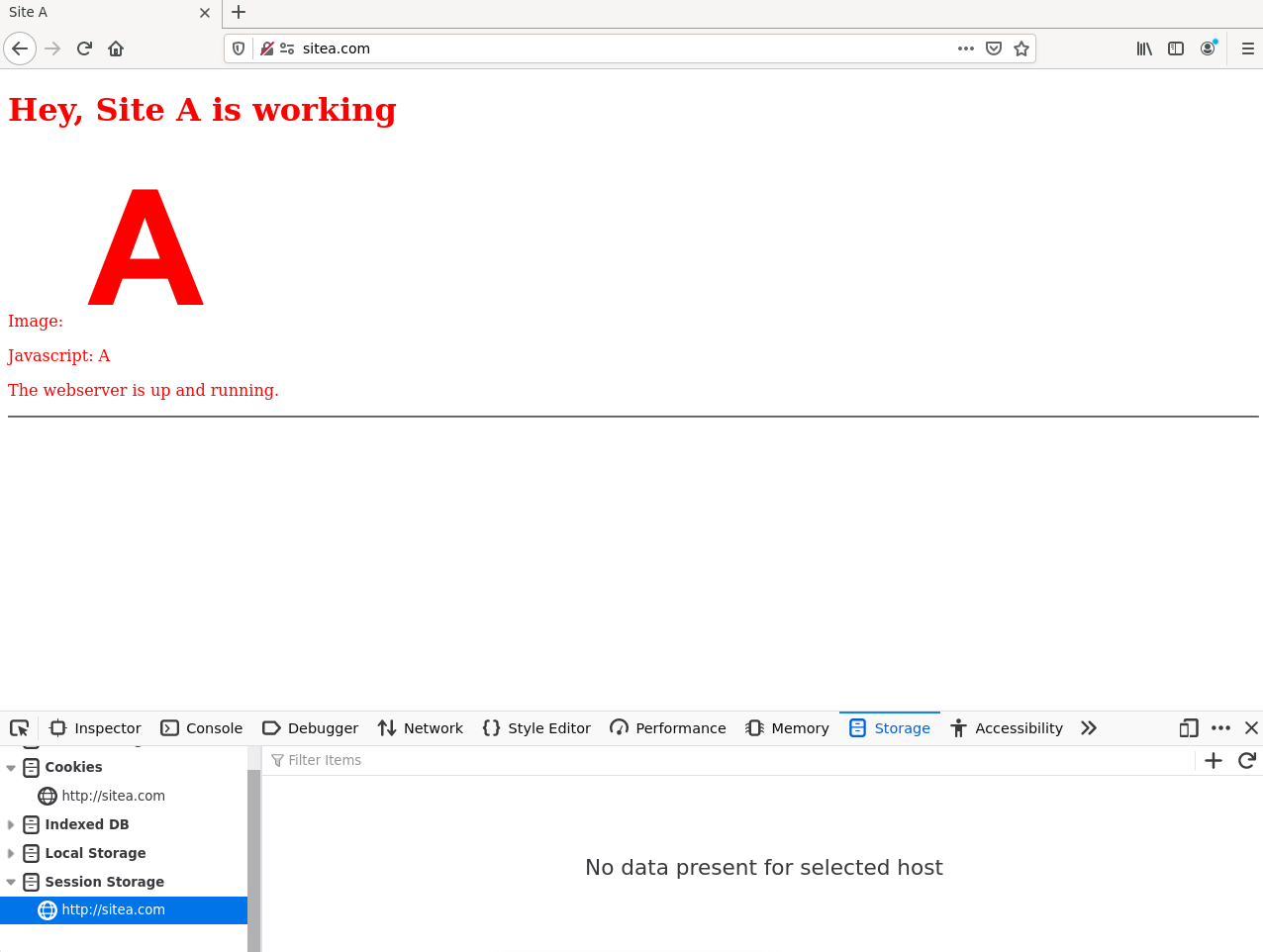
**localStorage** is a web storage mechanism that allows you to store data with no expiration, persisting across sessions.

* **Domain-specific Storage**: localStorage is tied to the domain that sets it. Only pages from the **same domain** can access the stored data. If a different domain tries to access it, they won’t see anything. This ensures that localStorage is isolated per domain for security and privacy reasons.
* **No Expiration**: Unlike sessionStorage, data in localStorage does not expire. It stays stored even after the browser is closed and reopened. The data remains until it's either manually deleted by the user (through browser settings) or programmatically cleared by the site.
* **Accessible Across Tabs**: Data stored in localStorage is accessible from any tab or window under the same domain. So, if you open multiple tabs or windows of the same domain, they can all share the data stored in localStorage. This makes it ideal for things like user settings or data that should be shared between multiple sessions.
* **Persistent Storage**: Because localStorage has no expiration, it’s useful for storing data that you want to keep around, like user preferences, saved shopping carts, or authentication tokens (though sensitive data like passwords should never be stored in localStorage).

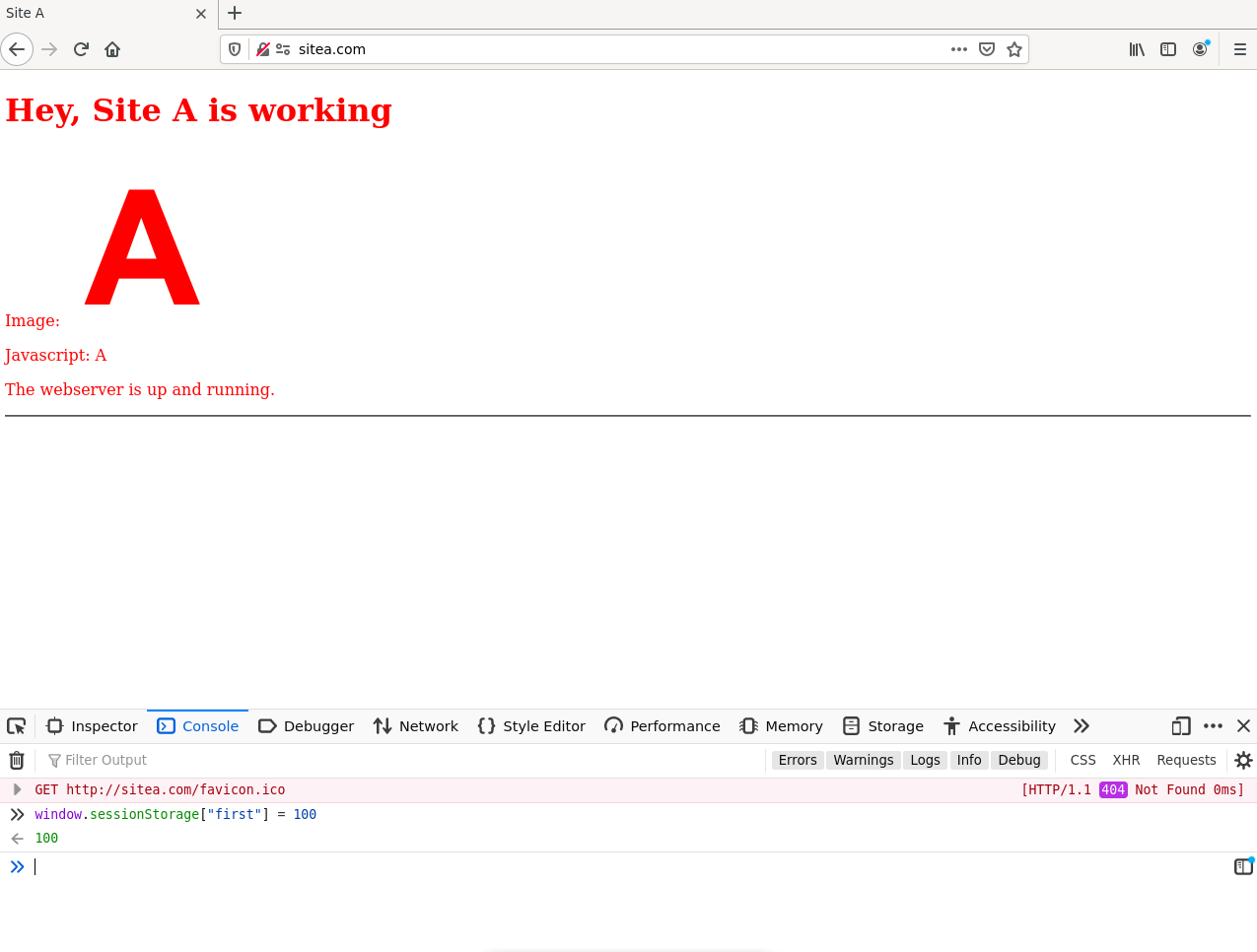
**1.6.2 Session Storage**

**Load www.sitea.com in one tab, open Web developer panel > Storage, check that the**

**Session Storage is empty (Storage >> Session Storage).**

****

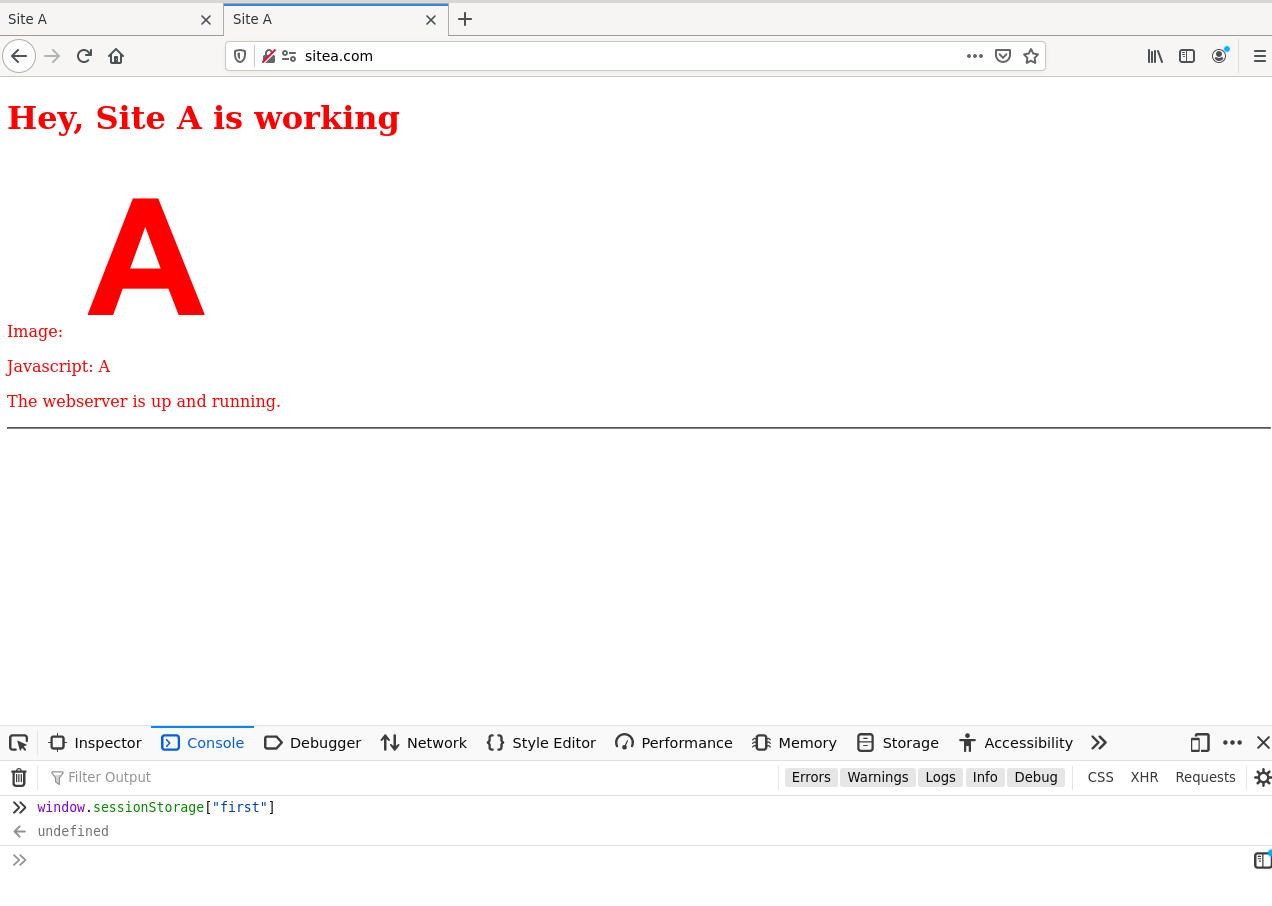
**a) Define a new variable named first = 10.**

****

**b) Open sitea.com in a new tab. Can you access first variable in this tab? Change first to a**

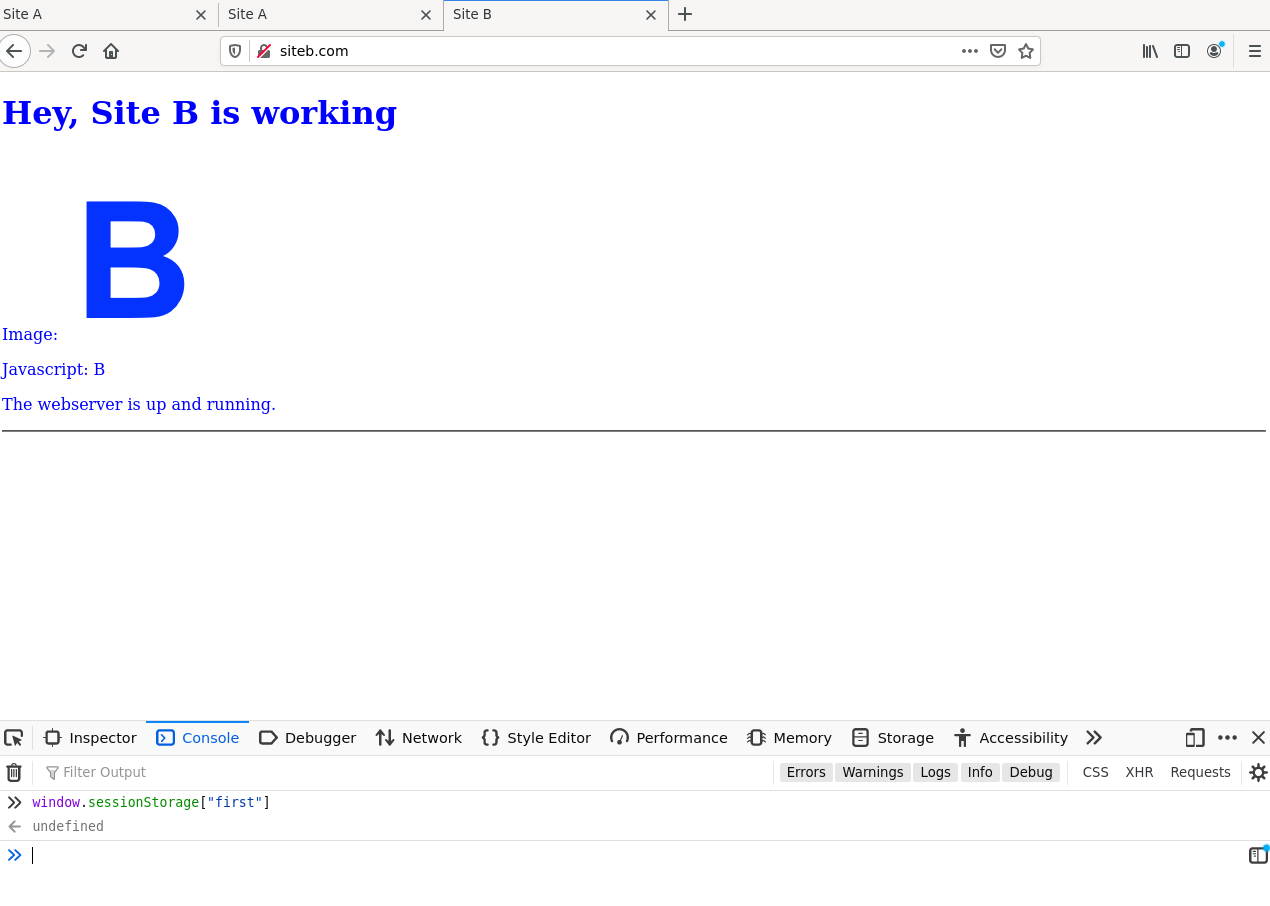
**new value, go back to the previous sitea.com tab, what is the value of first then.**

I cant access first variable in the new tab

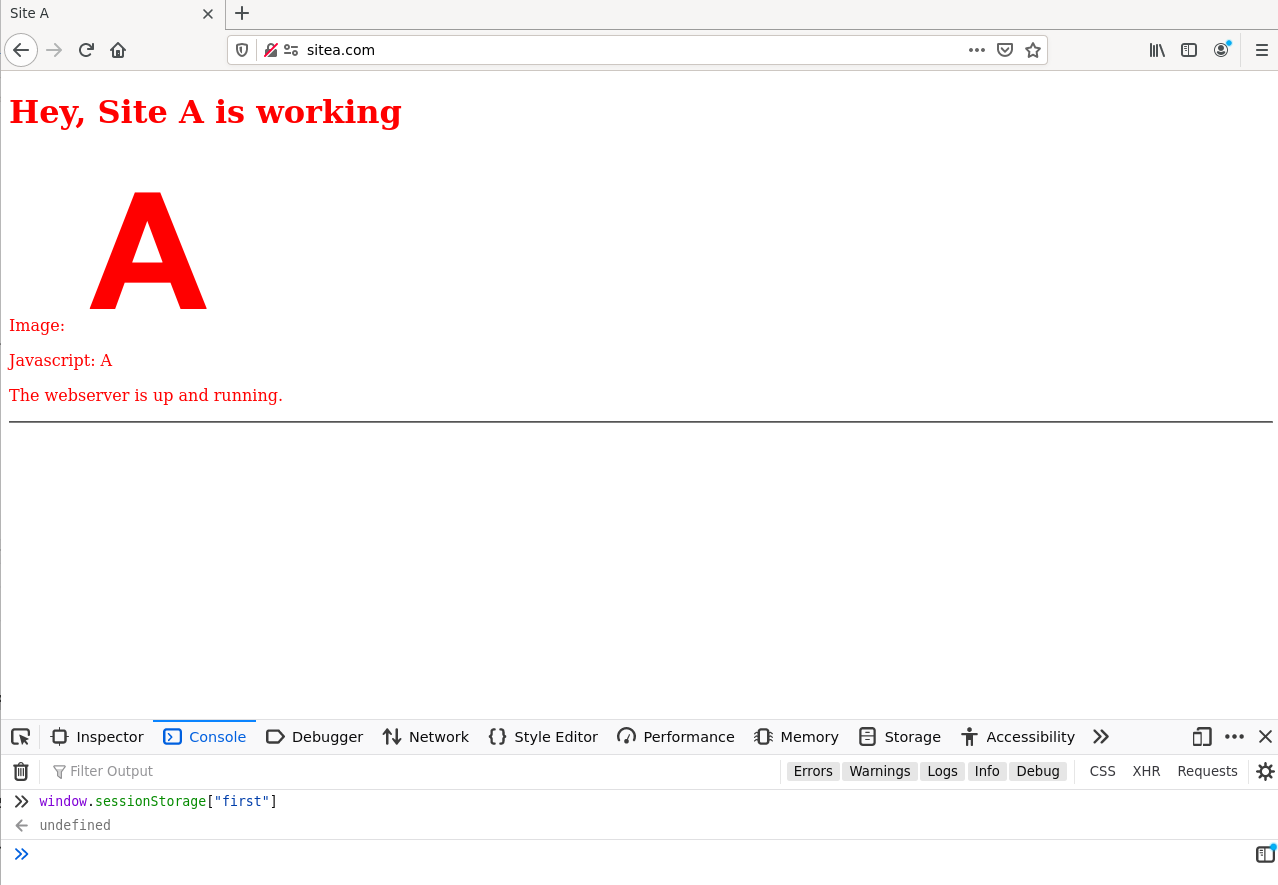


**c) Can you access first from siteb.com?**

I also cant access first variable in siteb.com

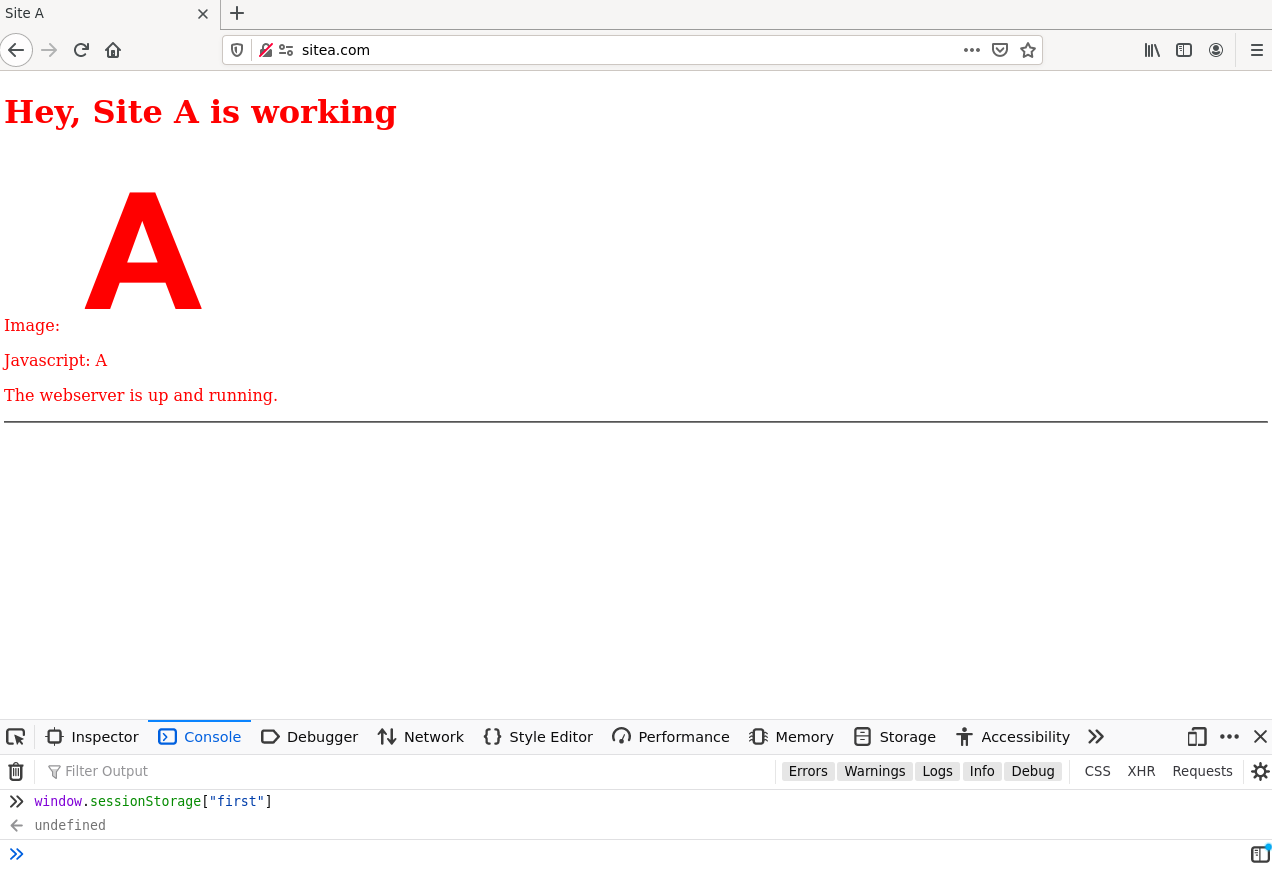


**d) Close all sitea.com tabs. How about first variable when sitea.com is re-opened.**

****

The variable when sitea.com is re-opened cannot access

**e) Close browser then re-open, does first variable still exist?**

****

No, the variable does not exists

**f) What can you conclude about Session Storage?**

Session Storage is a web storage mechanism that allows you to store data for the duration of a single browser session. Here's a more detailed explanation:

* **Tab-specific Storage**: Session Storage is scoped to the tab or window where it's created. This means that if you store a value in one tab, it won't be accessible from a different tab, even if both tabs are on the same domain. It's like having a temporary space that's unique to each tab or window.
* **Same Domain**: Data in Session Storage is associated with the domain that created it, but it’s still specific to the individual tab or window. So, opening a new tab with the same domain won’t allow you to access the stored data from the previous tab.
* **Lifetime of the Session**: The data persists only for the duration of the page session, which lasts as long as the tab or window is open. Once the tab or window is closed, the Session Storage is cleared automatically. Even if you close and reopen the browser, the session data won't be available anymore.