FIND THE FIX

**1. uncaught typeerror : Cannot read property**

It is a very common error when working with object and array to get a **TypeError**:C**annot read property** 'name' of undefined . This happens when we try to access a **property** of a value that is undefined or null in JavaScript

If accessing the element that is not there, it will return undefined. If we are accessing any element of that undefined value it will return this error



## 2. TypeError: ‘undefined’ is not an object (evaluating

This is an error that occurs in Safari when you read a property or call a method on an undefined object. You can test this very easily in the Safari Developer Console. This is essentially the same as the above error for Chrome, but Safari uses a different error message.



## TypeError: null is not an object (evaluating

This is an error that occurs in Safari when you read a property or call a method on a null object. You can test this very easily in the Safari Developer Console.



Interestingly, in JavaScript, null and undefined are not the same, which is why we see two different error messages. Undefined is usually a variable that has not been assigned, while null means the value is blank. To verify they are not equal, try using the strict equality operator:

## 4. (unknown): Script error

The Script error occurs when an uncaught JavaScript error crosses domain boundaries in violation of the cross-origin policy. For example, if you host your JavaScript code on a CDN, any uncaught errors (errors that bubble up to the window.onerror handler, instead of being caught in try-catch) will get reported as simply "Script error" instead of containing useful information. This is a browser security measure intended to prevent passing data across domains that otherwise wouldn’t be allowed to communicate.

To get the real error messages, do the following:

**1. Send the Access-Control-Allow-Origin header**

Setting the Access-Control-Allow-Origin header to \* signifies that the resource can be accessed properly from any domain. You can replace \* with your domain if necessary: for example, Access-Control-Allow-Origin: www.example.com. However, handling multiple domains gets tricky, and may not be worth the effort if you’re using a CDN due to caching issues that may arise. See more **here**.

Here are some examples on how to set this header in various environments:

**Apache**

In the folders where your JavaScript files will be served from, create an .htaccess file with the following contents:

Header add Access-Control-Allow-Origin "\*"

**Nginx**

Add the add\_header directive to the location block that serves your JavaScript files:

location ~ ^/assets/ {

add\_header Access-Control-Allow-Origin \*;

}

**HAProxy**

Add the following to your asset backend where JavaScript files are served from:

rspadd Access-Control-Allow-Origin:\ \*

**2. Set crossorigin="anonymous" on the script tag**

In your HTML source, for each of the scripts that you’ve set the Access-Control-Allow-Origin header for, set crossorigin="anonymous" on the SCRIPT tag. Make sure you verify that the header is being sent for the script file before adding the crossorigin property on the script tag. In Firefox, if the crossorigin attribute is present but the Access-Control-Allow-Origin header is not, the script won’t be executed.

## 5. TypeError: Object doesn’t support property

This is an error that occurs in IE when you call an undefined method. You can test this in the IE Developer Console.



## 6. TypeError: ‘undefined’ is not a function

This is an error that occurs in Chrome when you call an undefined function. You can test this in the Chrome Developer Console and Mozilla Firefox Developer Console.



## 7. Uncaught RangeError

This is an error that occurs in Chrome under a couple of circumstances. One is when you call a recursive function that does not terminate. You can test this in the Chrome Developer Console.



var a = new Array(4294967295); //OK

var b = new Array(-1); //range error

var num = 2.555555;

document.writeln(num.toExponential(4)); //OK

document.writeln(num.toExponential(-2)); //range error!

num = 2.9999;

document.writeln(num.toFixed(2)); //OK

document.writeln(num.toFixed(105)); //range error!

num = 2.3456;

document.writeln(num.toPrecision(1)); //OK

document.writeln(num.toPrecision(0)); //range error!

## 8. TypeError: Cannot read property ‘length’

This is an error that occurs in Chrome because of reading length property for an undefined variable. You can test this in the Chrome Developer Console.



## 9. Uncaught TypeError: Cannot set property

When we try to access an undefined variable it always returns undefined and we cannot get or set any property of undefined. In that case, an application will throw “Uncaught TypeError cannot set property of undefined.”



## 10. ReferenceError: event is not defined

This error is thrown when you try to access a variable that is undefined or is outside the current scope. You can test it very easily in Chrome browser.

