1. Uncaught TypeError: Cannot read property

If you’re a JavaScript developer, you’ve probably seen this error more than you care to admit. This one occurs in Chrome when you read a property or call a method on an undefined object. You can test this very easily in the Chrome Developer Console.

Var foo;

Undefined

Foo.bar

Uncaught error

There are two important things realize here:

1. A component’s state (e.g. this.state) begins life as undefined.
2. When you fetch data asynchronously, the component will render at least once before the data is loaded – regardless of whether it’s fetched in the constructor, componentWillMount or componentDidMount. When Quiz first renders, this.state.items is undefined. This, in turn, means ItemList gets items as undefined, and *you* get an error – "Uncaught TypeError: Cannot read property ‘map’ of undefined" in the console.

This is easy to fix. The simplest way: Initialize state with reasonable default values in the constructor.

## 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.

var testArray=undefined;

if(testArray.length===0){

console.log(“Array is empty”);

}

Uncaught SyntaxError: Invalid or unexpected token

## 3. 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.

var testArray=undefined;

if(testArray.length===0){

console.log(“Array is empty”);

}

Uncaught SyntaxError: Invalid or unexpected token

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:

Undefined==null

True

Undefined===null

False

One way this error might occur in a real world example is if you try using a DOM element in your JavaScript before the element is loaded. That’s because the DOM API returns null for object references that are blank.

## 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**](http://stackoverflow.com/questions/1653308/access-control-allow-origin-multiple-origin-domains).

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\*;

**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.

this.isAwesome()

object doesn’t support property or method ‘iAwesome’

This is equivalent to the error "TypeError: ‘undefined’ is not a function" in Chrome. Yes, different browsers can have different error messages for the same logical error.

This is a common problem for IE in web applications that employ JavaScript namespacing. When this is the case, the problem 99.9% of the time is IE’s inability to bind methods within the current namespace to the this keyword. For example, if you have the JS namespace Rollbar with the method isAwesome. Normally, if you are within the Rollbar namespace you can invoke the isAwesome method with the following syntax:

Chrome, Firefox and Opera will happily accept this syntax. IE, on the other hand, will not. Thus, the safest bet when using JS namespacing is to always prefix with the actual namespace.

## 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.

This.foo()

Uncaught TypeError: this.foo is not a function

As JavaScript coding techniques and design patterns have become increasingly sophisticated over the years, there’s been a corresponding increase in the proliferation of self-referencing scopes within callbacks and closures, which are a fairly common source of this/that confusion.

Alternatively, in the newer browsers, you can use the bind() method to pass the proper reference:

document.addEventListener("click",this.clearBoard.bind(this));

## 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(1);

function recurese(a){

a[0]=newArray(1);

recurse(a[0]);

}

uncaught ReferenceError: array is not defined

at <anonymous>:1:7

(anonymous) @ VM189:1

It may also happen if you pass a value to a function that is out of range. Many functions accept only a specific range of numbers for their input values. For example, Number.toExponential(digits) and Number.toFixed(digits) accept digits from 0 to 100, and Number.toPrecision(digits) accepts digits from 1 to 100.

## 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.

var mybutton=undefined

Undefined

Mybutton.length

Uncaught TypeError: Cannot read property ‘length’ of undefined

At<anonymous>:1:10

## 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.”

var test=undefined

undefined

test.value=0

Uncaught TypeError: cannot set the property ‘value’ of undefined

at <anonymous>

For example, in the Chrome browser:

If the test object does not exist, error 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.