Types of errors and fixes

Uncaught TypeError: Cannot read property:

This one occurs in Chrome when you read a property or call a method on an undefined object. This can occur for many reasons, but a common one is improper initialization of state while rendering the UI components.

Fix: Initialize the variables o properties with reasonable default values.

## 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.

Fix: Initialize the variables o properties with reasonable default values.

## 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. 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. 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.

Fix: Any JS code that executes and deals with DOM elements should execute after the DOM elements have been created. JS code is interpreted from top to down as laid out in the HTML. So, if there is a tag before the DOM elements, the JS code within script tag will execute as the browser parses the HTML page. You will get this error if the DOM elements have not been created before loading the script.

## (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.

Fix: 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:

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.

## 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 is equivalent to the error "TypeError: ‘undefined’ is not a function" in Chrome.

Fix: The safest bet when using JS namespacing is to always prefix with the actual namespace.

## 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.

Fix: A traditional, old-browser-compliant solution is to simply save your reference to this in a variable that can then be inherited by the closure. in the newer browsers, you can use the bind() method to pass the proper reference

## 

## 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. 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.

## TypeError: Cannot read property ‘length’

This is an error that occurs in Chrome because of reading length property for an undefined variable. You normally find length defined on an array, but you might run into this error if the array is not initialized or if the variable name is hidden in another context.

Fix: Remove parameters in the function declaration statement (it turns out you want to access those variables that are declared outside of the function, so you don’t need parameters for your function)

Invoke the function passing it the array that we declared.

## 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.”

## 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. If you’re getting this error when using the event handling system, make sure you use the event object passed in as a parameter. Older browsers like IE offer a global variable event, and Chrome automatically attaches the event variable to the handler. Firefox will not automatically add it. Libraries like jQuery attempt to normalize this behavior. Nevertheless, it’s best practice to use the one passed into your event handler function.

References: <https://rollbar.com/blog/top-10-javascript-errors/>