**1.uncaught typeerror cannot read property javascript error handling**

The simplest way to handle : Initialize state with reasonable default values in the constructor

function myFunc(inVar) {

if (inVar === undefined) {

console.log(inVar.not)

}

return inVar;

}

var testVar;

console.log(myFunc(testVar));

After error handling:

function test(t) {

if (t === undefined)

{ console.log(t.tt)

}

return t;

}

var a;

if(typeof(a) == 'undefined')

{

console.log("Please initialize value");

}

else

console.log(test(a)); //function call

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

This is an error that occurs when you read a property or call a method on an undefined object.

var arr=undefined;

if(arr.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 when you read a property or call a method on a null object.

var arr=undefined;

if(arr.length===0){

console.log(“Array is empty”);

}

Uncaught SyntaxError: Invalid or unexpected token

Null and undefined are not same in JavaScript. Null means that the object has a blank value. Undefined means a variable that is not assigned. Using a strict equality operator can verify that they are not equal.

Undefined==null

True

Undefined===null

False

**Fix** – An event listener is the perfect solution for this type of errors. It notifies the developer once the page is ready. The init() method can use the DOM elements once the event listener is fired.

**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.

**Fix** – Try-catch should always be used to handle errors

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

This error usually occurs in Internet Explorer when an undefined method is called. It can be compared to the undefined function error that occur in Chrome. For web applications that use JavaScript namespacing, this error is quite common. IE cannot bind “this” keyword to the current namespace. For example, this.isAwesome() works properly in all browsers but throws an exception in Chrome**.**

object doesn’t support property or method ‘isAwesome’

**Fix** – When using namespacing, this error can be avoided by using the actual namespace as a prefix

**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

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 error occurs mostly in Chrome due to an undefined variable’s length property. Normally an array has its length defined. But when the variable name of an array remains hidden or if the array is not initialized, this error happens.

var mybutton=undefined

Undefined

Mybutton.length

Uncaught TypeError: Cannot read property ‘length’ of undefined

Fix – This error can be fixed in 2 ways

* In the statement where the function is declared, parameters should be removed.
* The function should be invoked in the array that is declared.

Most of the errors that occur when the browser compiles the JavaScript are either undefined or null type errors. If the developer uses the strict compiler in a static checking system like Typescript, these errors can be avoided. It will give the warning that a type is not defined but expected. Even if Typescript is not used, guard clauses can be used to check if the objects are undefined before they are used.

**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

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.