



FINDING SECRETS WITH THE JAVASCRIPT CONSOLE

WITH BB KING

CONSULTING PENTESTER, REPORT WRITER, INSTRUCTOR

@BBhacKing

<https://www.antisyphontraining.com/instructor/bb-king/>

MODERN WEBAPP PENTESTING I AND II

REPORTING FOR PENTESTERS

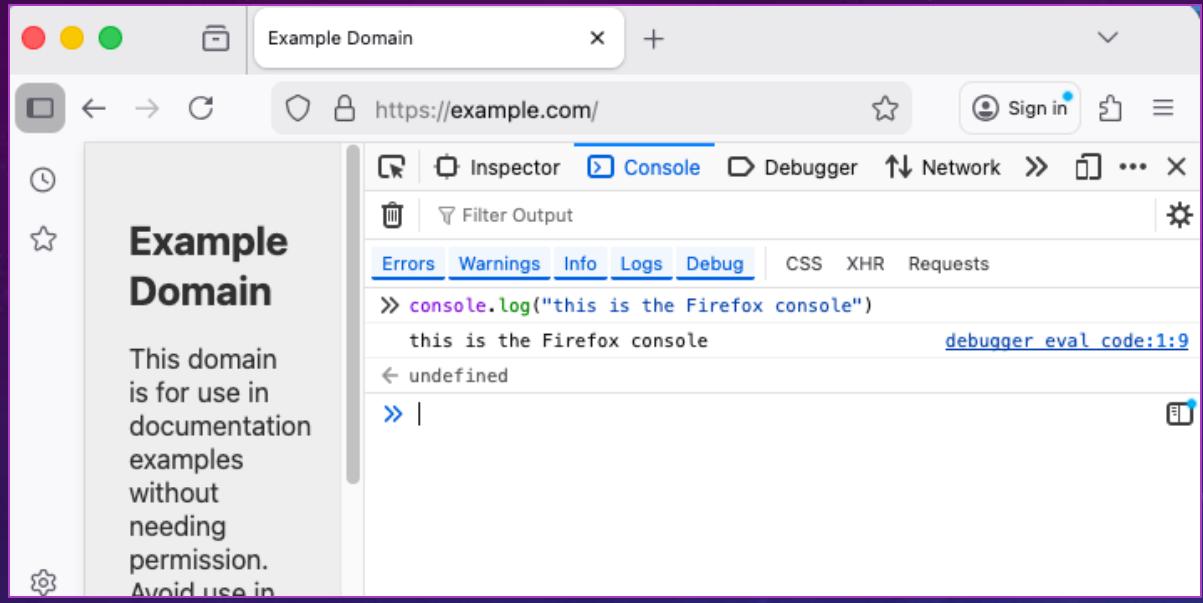
BURP SUITE WORKSHOP

- In-Person, Simulcast, and On Demand
- MWAP I at WWHF Mile High, February 2026
- <https://wildwesthackinfest.com/>

PLEASE PLAY ALONG

- Visit a site of your choice in the browser of your choice.
 - <https://example.com/> is always a nice example.
 - <https://www.rfc-editor.org/rfc/rfc6761.html#section-6.5>
 - <https://blackhillsinfosec.com/> is also a nice place.
 - Other sites available, too. Choose a quiet one.
- Open the console. (tap F12)
- Play there as we go.

DEVELOPER TOOLS / CONSOLE / F12



This screenshot shows the Firefox Developer Tools interface. The title bar says "Example Domain". The address bar shows "https://example.com/". The main content area displays the text "Example Domain" and a message: "This domain is for use in documentation examples without needing permission. Avoid use in". Below this, the Firefox Developer Tools Console tab is active, showing the following output:

```
» console.log("this is the Firefox console")
this is the Firefox console
debugger eval code:1:9
← undefined
» |
```

The "Console" tab is selected in the top navigation bar, and the "Debug" tab is active within the console panel. The bottom navigation bar shows tabs for Errors, Warnings, Info, Logs, and Debug, with "Info" being the active tab. A prominent warning message is displayed at the bottom:

⚠ Filter Output

Scam Warning: Take care when pasting things you don't understand. This could allow attackers to steal your identity or take control of your computer. Please type 'allow pasting' below (no need to press enter) to allow pasting.

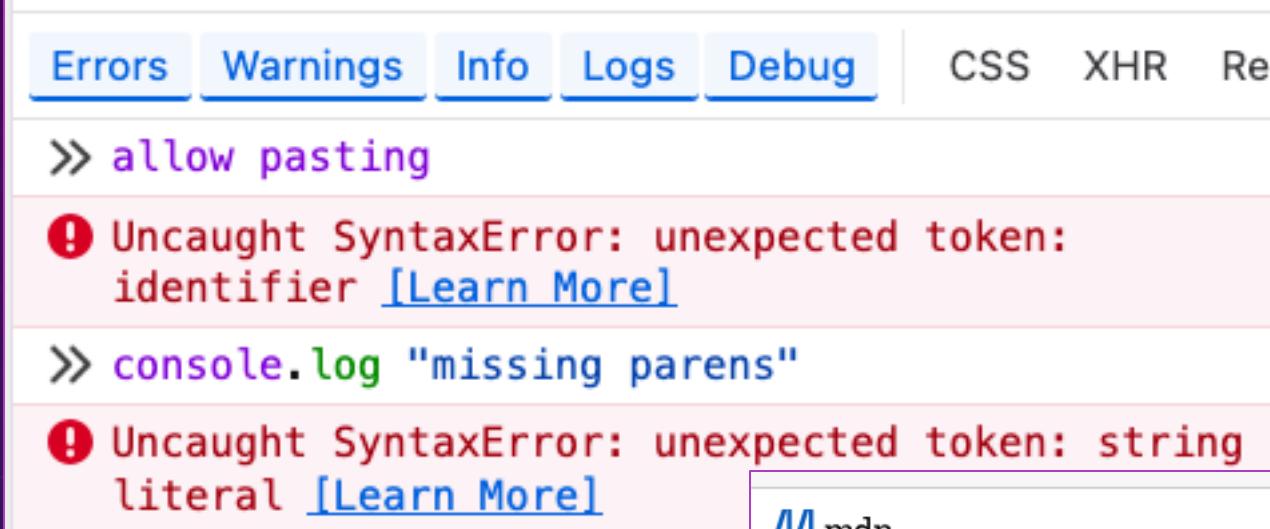


This screenshot shows the Chrome DevTools interface. The title bar says "Example Domain". The address bar shows "https://example.com". The main content area displays the text "Example Domain" and a message: "This domain is for use in documentation examples". Below this, the Chrome DevTools Console tab is active, showing the following output:

```
✗ Failed to load resource: the server responded with a status of 404 ()
> "this is the Chrome console"
< 'this is the Chrome console'
>
```

The "Console" tab is selected in the top navigation bar. The bottom navigation bar shows tabs for Elements, Console, Sources, and Network, with "Console" being the active tab. A message in the bottom right corner indicates "1 issue" and "No Issues".

DEVELOPER TOOLS / CONSOLE / F12



Errors Warnings Info Logs **Debug** CSS XHR Req

» `allow pasting`

! **Uncaught SyntaxError: unexpected token: identifier** [\[Learn More\]](#)

» `console.log "missing parens"`

! **Uncaught SyntaxError: unexpected token: string literal** [\[Learn More\]](#)

Clicking on [\[Learn More\]](#) may encourage learning.



MDN

Reference > JavaScript error reference > SyntaxError: Unexpected token

Theme English (US)

In this article

- Message
- Error type
- What went wrong?
- Examples

SyntaxError: Unexpected token

The JavaScript exceptions "unexpected token" occur when the parser does not see a token it recognizes at the given position, so it cannot make sense of the structure of the program. This might be a simple typo.

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors>

BUT FIRST: A BIT ABOUT JAVASCRIPT

❖ AI Overview

"But first" is a common phrase used to prioritize an action, most frequently having a coffee before doing something else. It can also be the name of a coffee brand or a book, like [But First, Coffee: A Guide to Brewing from the Kitchen to the Bar](#), and is sometimes used in marketing to create a strong association between coffee and a prioritized activity.



STATEMENTS

“one or more lines that represent an action”

- or -

“Something that ends with a semi-colon” (mostly)

You can omit the semi-colon, but don’t do that.

- Case-sensitive
- (Amount-of-)whitespace-insensitive

...so at least it's not Python

```
owner = "John";  
let keep_going = true;  
let use_var_instead = false;
```

CONTROL FLOW

Conditionals: if / then (implied) / else

```
>> if ( true ){
    console.log( "Yup." );
}
Yup.
```

```
>> if( top.location.href == "Ohio"){
    console.log("woah. Ohio");
} else {
    console.log("that's not how top.location works");
}
that's not how top.location works
```

<https://web.dev/learn/javascript/control-flow>

COMPARISON OPERATORS

=	assignment	
==	loose equality	(same value? ...maybe after <u>coercion</u> ?)
===	strict equality	(straight up same object.)
!	not	(negate the equalities: !=, !==)
>	greater than	
>=	greater than or equal to	
<	less than	
<=	less than or equal to	

BEATING UP ON JAVASCRIPT: AN INTRODUCTION

- Truthy and Falsy
 - Formalized Truthiness
- Type coercion
 - *Coerces LHS and RHS to a common datatype, then compares the results.*

```
» two = "2"  
← "2"  
» also_two = 2  
← 2  
» two == also_two  
← true  
» two === also_two  
← false
```

```
» "2" = 22 - 20  
! Uncaught SyntaxError: invalid assignment left-hand side [Learn More]  
» "2" == 22 - 20  
← true  
» "2" == "22" - "20"  
← true  
» "2" == "22 - 20"  
← false
```

TRUTHINESS TYPE A

==== STRICT EQUALITY ====

“as found”
(no coercion)

`a === b`

“are these the same item?”

true	true	false	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	undefined	null	NaN
false	1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	undefined	null	NaN		
1	0	-1	"true"	"false"	"1"	"0"	"-1"	""	undefined	null	NaN			
0	-1	"true"	"false"	"1"	"0"	"-1"	""	undefined	null	NaN				
-1	"true"	"false"	"1"	"0"	"-1"	""	undefined	null	NaN					
"true"	"false"	"1"	"0"	"-1"	""	undefined	null	NaN						
"false"	"1"	"0"	"-1"	""	undefined	null	NaN							
"1"	"0"	"-1"	""	undefined	null	NaN								
"0"	"-1"	""	undefined	null	NaN									
"-1"	""	undefined	null	NaN										
""	undefined	null	NaN											
undefined	null	NaN												
null	NaN													
NaN														

TRUTHINESS TYPE B

== LOOSE EQUALITY ==

“as interpreted” (with coercion)

$$a == b$$

“do these have the same value?”

CONTEMPLATING THE IFS

...with coercion.

<https://dorey.github.io/JavaScript-Equality-Table/>

A standard IF statement. If(*value*) /*- green -*/ else { /*- white -*/ }

Note: This row does not match up with any of the rows in the other table.

true		if (true) { /* executes */ }
false		if (false) { /* does not execute */ }
1		if (1) { /* executes */ }
0		if (0) { /* does not execute */ }
-1		if (-1) { /* executes */ }
"true"		if ("true") { /* executes */ }
"false"		if ("false") { /* executes */ }
"1"		if ("1") { /* executes */ }
"0"		if ("0") { /* executes */ }
"-1"		if ("-1") { /* executes */ }
""		if ("") { /* does not execute */ }
null		if (null) { /* does not execute */ }
undefined		if (undefined) { /* does not execute */ }
Infinity		if (Infinity) { /* executes */ }
-Infinity		if (-Infinity) { /* executes */ }
[]		if ([]){ /* executes */ }
{}		if ({}){ /* executes */ }
[[]]		if ([[]]) { /* executes */ }
[0]		if ([0]) { /* executes */ }
[1]		if ([1]) { /* executes */ }
NaN		if (NaN) { /* does not execute */ }



COMPARISON-ISH OPERATORS

```
> myWord = 'homeowner';
< 'homeowner'
> currencies = ['USD', 'EUR', 'YEN'];
< ▶ (3) ['USD', 'EUR', 'YEN']
```

```
myWord.indexOf('meow'); // is 'meow' in that String?
// Where?
```

```
> myWord.indexOf('meow');
< 2
```



```
currencies.includes('YEN');
// is 'YEN' in this Array?
// True/False.
```

```
> currencies.includes('YEN');
< true
```

HOW THE AI WANTED ME TO SAY THAT



MYWORD = 'HOMEOWNER';



CURRENCIES =
['USD','EUR','YEN'];



MYWORD.INDEXOF('MEOW')
IS 'MEOW' IN THAT
STRING? WHERE?



CURRENCIES.INCLUDES('YEN')
IS 'YEN' IN THIS
ARRAY? T/F

PROTOTYPAL INHERITANCE

(SORRY I DON'T MAKE THE NAMES)

- A value inherits the properties and methods of its constructor...
 - ...even if it hasn't been constructed yet.
- The String class has a method called `toUpperCase()`
- You can call this on a string literal (which is not a String).

```
» "Passw0rd1".toUpperCase()  
← "PASSW0RD1"
```

PROTOTYPAL INHERITANCE

(STILL SORRY)

```
» "InTerESTinGworD".toLowerCase() == "INTERESTINGWORD".toLowerCase()  
← true
```

Handy for on-the-fly comparisons.

```
» [1,2,3].includes(2)  
← true
```

LOOPS
ARE WHERE
IT'S AT.



PROCESSING A “LIST” LIKE GRANDPA DID

for(start; stop; step){}

```
1 let str = "";
2
3 for (let i = 0; i < 9; i++) {
4     str += i;
5 }
6
7 console.log(str);
8 // Expected output: "012345678"
```

Canonical Use Case...

...Gets Clunky With Arrays.

```
1 const array = ["a", "b", "c"];
2
3 for (let i = 0; i < array.length; i++) {
4     console.log( array[i] )
5 }
6
7 // Expected output: "a"
8 // Expected output: "b"
9 // Expected output: "c"
```

PROCESSING ARRAYS WITHOUT INDEXING

for (var_name of iterable){}



```
1 const array = ["a", "b", "c"];
2
3 for (const element of array) {
4     console.log(element);
5 }
6
7 // Expected output: "a"
8 // Expected output: "b"
9 // Expected output: "c"
```

what is an "iterable"?

An iterable is any object that you can loop over one element at a time, like in a for-loop. [pythonlikeyoumeanit +1](#)

General idea

- Informally, if you can do "for each element in X: ...", then X is an iterable. [youtube](#) [reddit](#)
- Typical examples are lists, tuples, strings, dictionaries, sets, and many similar container-like objects. [stackoverflow +1](#)

perplexity.ai : Not Wrong

CONSOLE TRICKS - IDENTIFY LINKS

Put a blue outline around each link:

```
for( link of document.links ) {  
  link.style = "border: 5px solid blue";  
}
```

CONSOLE TRICKS - COLLECT LINKS

Make a list: Links That Go Offsite

https://en.wikipedia.org/wiki/Main_Page

```
for(link of document.links) {  
  if(link.href.indexOf('en.wikipedia') == -1){  
    console.log(link.href);  
  }  
}
```

CONSOLE TRICKS - COLLECT IDENTIFIERS

Make a list: Each Image's URL

```
const pic_urls = [];

for(pic of document.images){

  pic_urls.push(pic.src);

}

console.log(pic_urls);
```

```
» const pic_urls = [];
  for(pic of document.images){
    pic_urls.push(pic.src);
  }
  console.log(pic_urls);
← undefined

▼ (94) [...]
  0: "https://combo.staticflickr.com/pw/images/tour/en-us/create-accou
  1: "https://farm3.staticflickr.com/2853/buddyicons/60065287@N00_r.jp
  2: "https://farm1.staticflickr.com/932/buddyicons/36521981547@N01_r.
  3: "https://farm8.staticflickr.com/7317/buddyicons/15721381@N00_r.jp
  4: "https://farm9.staticflickr.com/8463/buddyicons/20244787@N07_r.jp
  5: "https://farm1.staticflickr.com/892/buddyicons/11797544@N05_r.jpg
  6: "https://farm6.staticflickr.com/5037/buddyicons/78522870@N07_r.jp
  7: "https://farm1.staticflickr.com/873/buddyicons/91720879@N02_r.jpg
  8: "https://farm8.staticflickr.com/7386/buddyicons/63652802@N02_r.jp
  9: "https://farm4.staticflickr.com/3718/buddyicons/26794697@N03_r.jp
  10: "https://farm66.staticflickr.com/65535/buddyicons/66422871@N00_r.jp
```

CONSOLE TRICKS – YOUR TURN

Make a list:

All the URLs the current page pulls scripts from.

PROCESSING A “LIST” OF OBJECT PROPERTIES

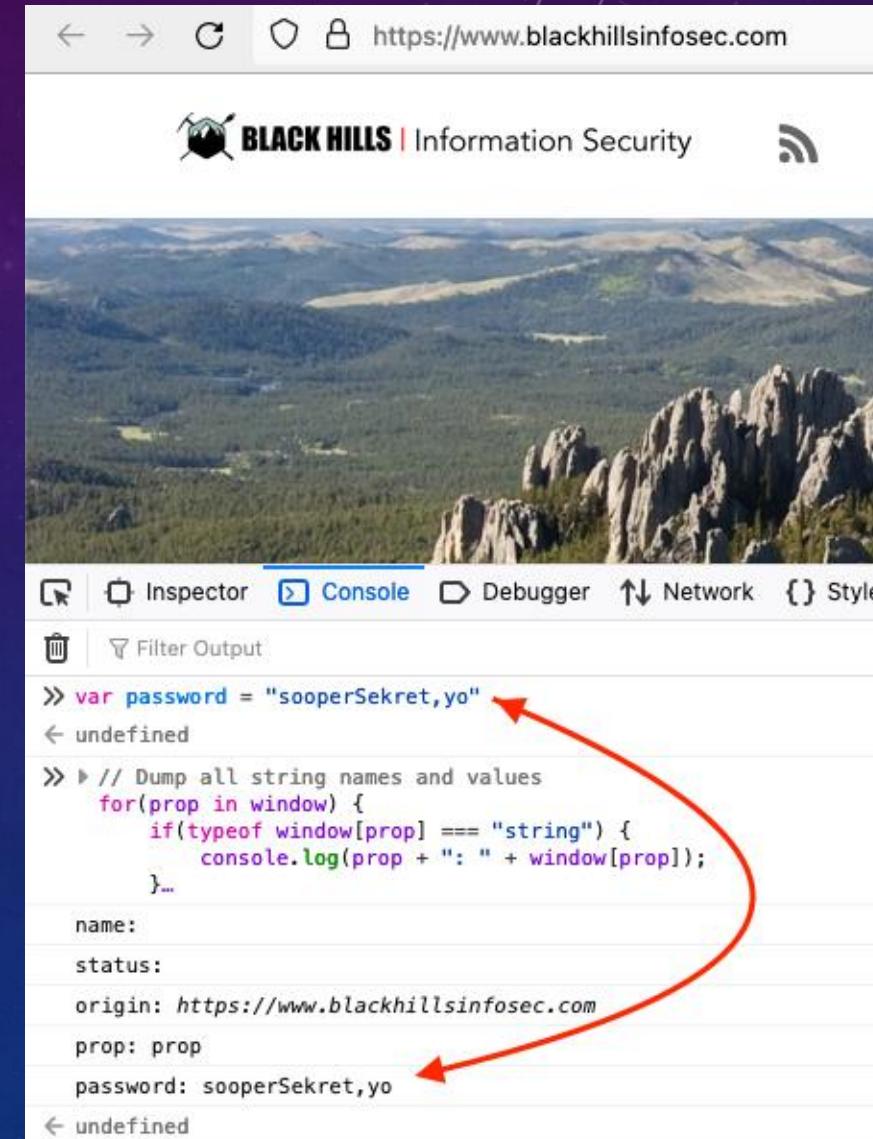
for (var_name in obj_name){}

```
1 const object = { a: 1, b: 2, c: 3 };
2
3 for (const property in object) {
4   console.log(` ${property}: ${object[property]}`);
5 }
6
7 // Expected output:
8 // "a: 1"
9 // "b: 2"
10 // "c: 3"
11
```

READING VARIABLES YOU DON'T KNOW ABOUT

Find variables and objects in the DOM

```
// Dump all string names and values
// Courtesy of Sean Verity, BHIS tester
for(prop in window) {
    if(typeof window[prop] === "string") {
        console.log(prop + ": " + window[prop]);
    }
}
// see also boolean, object, function, number
```



READING FROM SENSITIVE AREAS: COOKIES

```
» document.cookie
```

```
← "GeoIP=US:TX:; NetworkProbeLimit=0.001;  
VEECid=d4c06d8faa991578dcb8; enwikimwuser-sessionId=a00b6b0fb70494525068"
```

This cookie has four cookies.

But ... still just “cookie”

```
» document.cookie[0]
```

```
← "G"
```

```
» document.cookie(0)
```

! ▶ **Uncaught TypeError: document.cookie is not a function**
<anonymous> debugger eval code:1
[\[Learn More\]](#)

```
» typeof(document.cookie)
```

```
← "string"
```

READING FROM SENSITIVE AREAS: COOKIES

```
» document.cookie
```

```
← "GeoIP=US:TX:; NetworkProbeLimit=0.001;  
VEECid=d4c06d8faa991578dcb8; enwikimwuser-sessionId=a00b6b0fb70494525068"
```

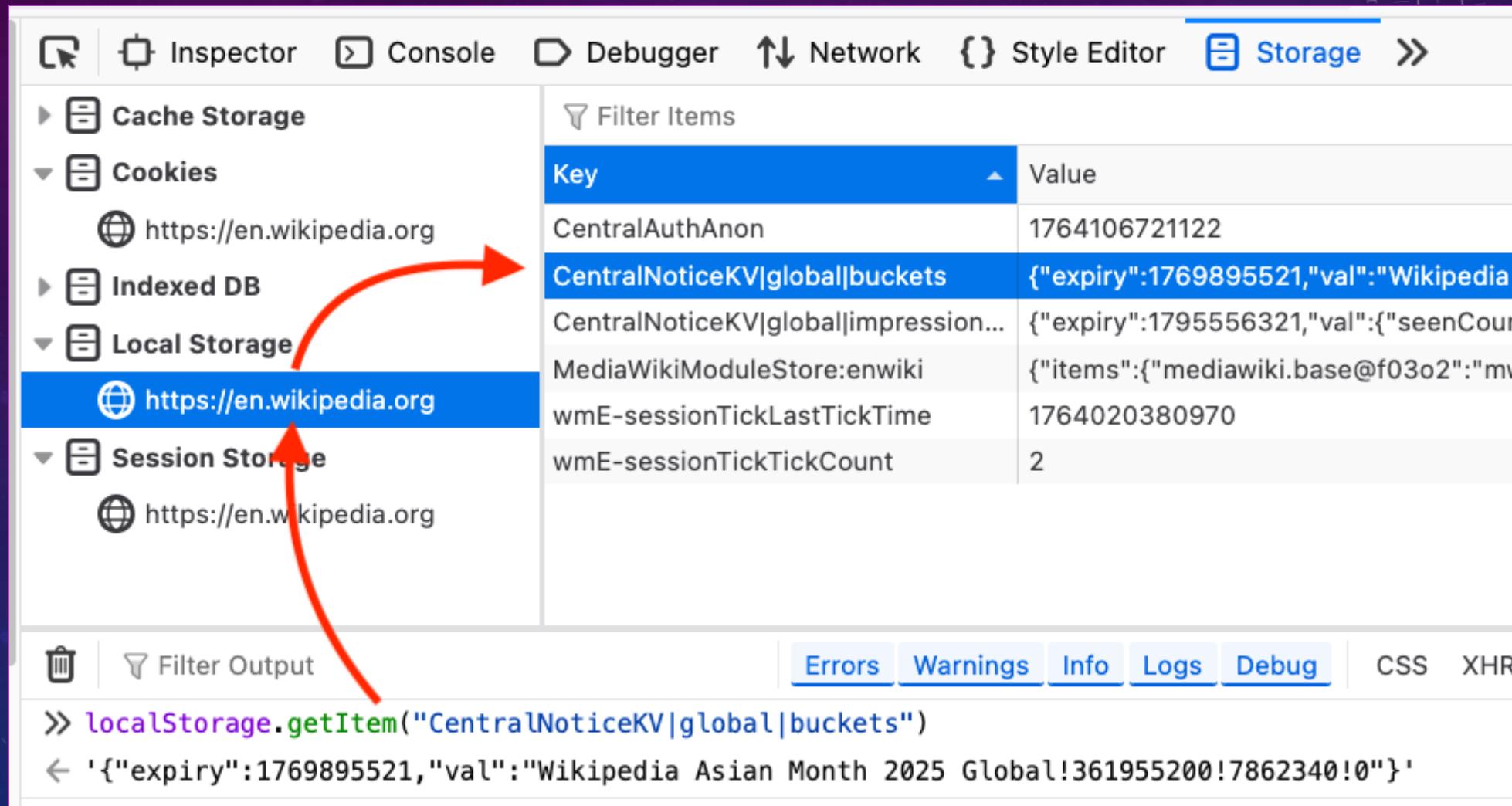
This cookie has four cookies.

Use `String.split('; ')` to separate them. If you want to do that..

```
» document.cookie
```

```
← "GeoIP=US:TX:; NetworkProbeLimit=0.001;  
VEECid=90f8f94054324f36e370; enwikimwuser-sessionId=28c8b8e67c250e63171e"  
» document.cookie.split('; ')[2]  
← "VEECid=90f8f94054324f36e370"
```

READING FROM SENSITIVE AREAS: DOM STORAGE



The screenshot shows the Storage tab of a browser developer tools interface. The left sidebar lists storage types and their contents:

- Cache Storage
- Cookies
- Indexed DB
- Local Storage
 - https://en.wikipedia.org
- Session Storage
 - https://en.wikipedia.org

Red arrows point from the "Local Storage" and "Session Storage" sections to the table on the right, indicating the focus of the analysis.

The main area displays a table of stored items:

Key	Value
CentralAuthAnon	1764106721122
CentralNoticeKV global buckets	{"expiry":1769895521,"val":"Wikipedia Asian Month 2025 Global!361955200!7862340!0"}
CentralNoticeKV global impression...	{"expiry":1795556321,"val":{"seenCour...
MediaWikiModuleStore:enwiki	{"items":{"mediawiki.base@f03o2":"mv...
wmE-sessionTickLastTickTime	1764020380970
wmE-sessionTickTickCount	2

At the bottom, the developer tools command line shows the command used to retrieve the item from Local Storage:

```
localStorage.getItem("CentralNoticeKV|global|buckets")
```

The response is:

```
{"expiry":1769895521,"val":"Wikipedia Asian Month 2025 Global!361955200!7862340!0"}
```

WHAT IF YOU DON'T KNOW THE NAME OF THE KEY?

```
localStorage.key( 0 ); // This is a function(call).  
localStorage.key( 1 ); // It is not array[indexing].
```

```
localStorage.length; // ...this looks promising...
```

LISTING JUST THE KEYS

```
for (let i = 0; i < localStorage.length; i++) {  
  console.log( localStorage.key(i) );  
}
```

// Fancier, but the same:

```
Object.keys(localStorage).forEach(key => {  
  console.log(key);  
});
```

LISTING THE KEYS AND THE VALUES

```
for (let i = 0; i < localStorage.length; i++) {  
  console.log( localStorage.getItem( localStorage.key(i) ) );  
}
```

KEYS AND VALUES: EASIER

```
console.table(localStorage);
```

https://developer.mozilla.org/en-US/docs/Web/API/console/table_static

CHAINING FUNCTIONS NOT DISCUSSED

```
» JSON.parse(localStorage.getItem("CentralNoticeKV|global|buckets"))
← ▶ Object { expiry: 1769895521, val: "Wikipedia Asian Month 2025 Global!361955200!
7862340!0" }
» JSON.parse(localStorage.getItem("CentralNoticeKV|global|buckets")).val
← "Wikipedia Asian Month 2025 Global!361955200!7862340!0"
```

CHAINING FUNCTIONS NOT DISCUSSED

```
> 'one,two,three'.split(',')
< ▶ (3) ['one', 'two', 'three']
```

split() breaks up a string...

```
> 'one,two,three'.split(',')[1]
< 'two'
```

...and array indexing is a thing.

atob() is base 64 decoding, for some reason.

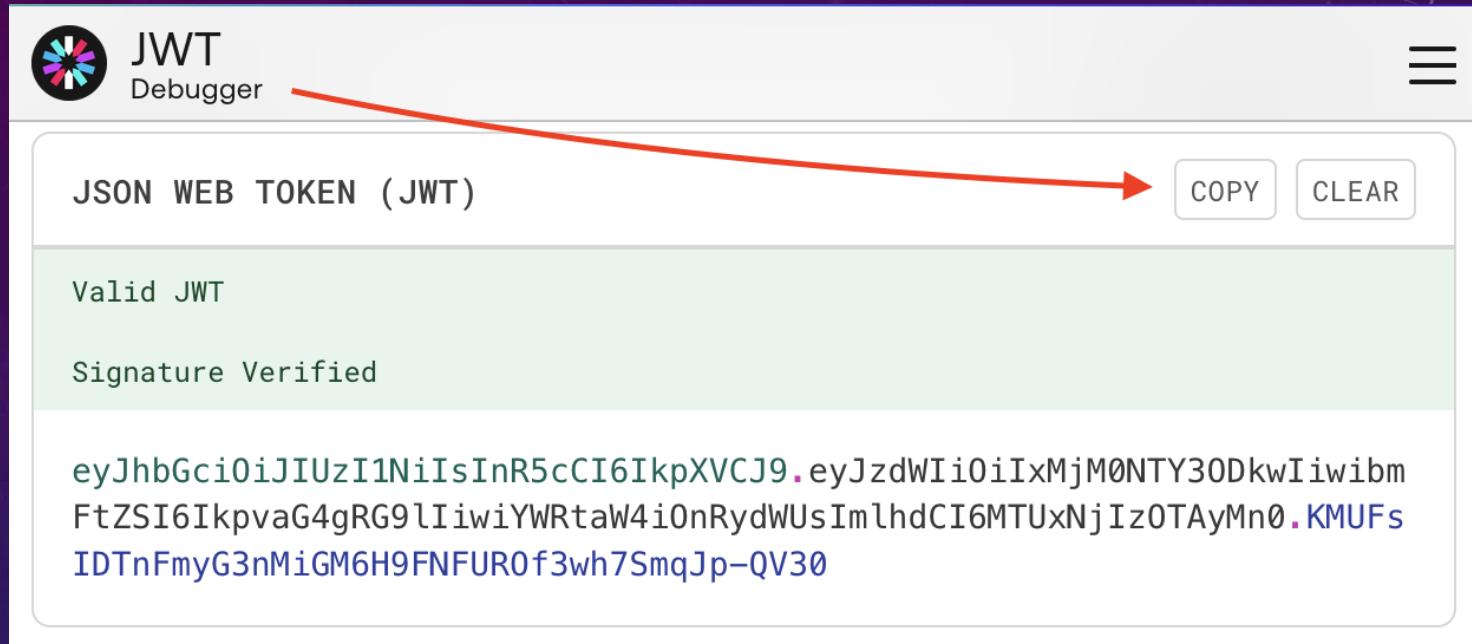
JSON.parse() from the last slide.

And dot notation, too.

These let us extract and decompose a JWT in just one statement!

TO PLAY ALONG ON THE NEXT SLIDE

go to <https://jwt.io/>



Copy the JWT to your clipboard

In the console, while you're on example.com, paste it in place of the word **HERE** in...

```
localStorage.setItem('jwt','HERE');
```



CHAINING FUNCTIONS: EXAMINE A JWT IN LOCAL STORAGE

```
» localStorage.getItem('jwt')
← "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZ
RG9lIiwiYWRtaW4iOnRydWUsImlhCI6MTUxNjIzOTAyMn0.KMUFsIDTnFmyG3nMiGM6H9I
» localStorage.getItem('jwt').split('.')
← ▶ Array(3) [ "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9",
  "eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWUsImlhCI6MTUxNjIzOTAyMn0.KMUFsIDTnFmyG3nMiGM6H9I
» localStorage.getItem('jwt').split('.')[1]
← "eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWUsImlhCI6MTUxNjIzOTAyMn0.KMUFsIDTnFmyG3nMiGM6H9I
» atob(localStorage.getItem('jwt').split('.')[1])
← '{"sub":"1234567890","name":"John Doe","admin":true,"iat":1516239022}'
» JSON.parse(atob(localStorage.getItem('jwt').split('.')[1]))
← ▶ Object { sub: "1234567890", name: "John Doe", admin: true,
  "iat": 1516239022 }
» JSON.parse(atob(localStorage.getItem('jwt').split('.')[1])).name
← "John Doe"
```

Questions?

MODERN WEBAPP
PENTESTING
I AND II

REPORTING FOR
PENTESTERS

BURP SUITE
WORKSHOP

- In-Person, Simulcast, and On Demand
- MWAP I at WWHF Mile High, February 2026
- <https://wildwesthackinfest.com/>
- <https://www.antisyphontraining.com/instructor/bb-king/>
- BB King | @BBhacKing

ANTISYPHON
TRAINING

POWERED BY RHN

WILD WEST
HACKIN' FEST
@Mile High

MODERN WEBAPP PENTESTING



BB KING



THIS NEXT BIT DOESN'T FIT ANYWHERE
BUT IT'S KIND OF FUN, SO ... "BONUS MATERIAL!"

IF YOU HATED ‘PROTOTYPAL INHERITANCE...’ YOU’LL LOVE... HOISTING AND THE TEMPORAL DAD ZONE

- ...It's Temporal *Dead Zone*, but I like the typo-ed version better.
- JS lets you declare variables anywhere.
 - So it is possible to use them before you declare them.
 - Doing that on purpose is not smart. Be smart.
- “Hoisting” is “pretending you didn’t do that”
 - Kind of. It does this by making a variable accessible anywhere in the scope where the variable is declared, even before the declaration.
- The “Temporal Dead Zone” is a result of this ~~silliness~~ flexibility.
 - It is: the “time” between the start of a block and when a “hoisted” variable is initialized.
 - [Learn More] <https://jsrocks.org/2015/01/temporal-dead-zone-tdz-demystified>

HOISTING AND THE TEMPORAL DAD ZONE

```
> x = 'outer scope';
(function(){
  console.log(x);
  x = 'inner scope';
})()
outer scope
```

Initialized too late.
Global variable wins.
Probably bad.

```
> x = 'outer scope';
(function(){
  console.log(x);
  var x = 'inner scope';
})()
undefined
```

Local Variable with var.
Hoisted but not Initialized.
Silently Undefined.
Probably not better.

```
> x = 'outer scope';
(function(){
  console.log(x);
  let x = 'inner scope';
})()
```

✖ ► Uncaught ReferenceError: Cannot access 'x' before initialization
at <anonymous>:3:17
at <anonymous>:5:2

Local Variable with let.
Hoisted but not Initialized.
Code won't run.
Problem solved. Right?

JUST DO IT RIGHT. THEN YOU DON'T HAVE TO REMEMBER ALL THAT.

```
> x = 'outer scope';
(function(){
  x = 'inner scope';
  console.log(x);
}())
inner scope
```

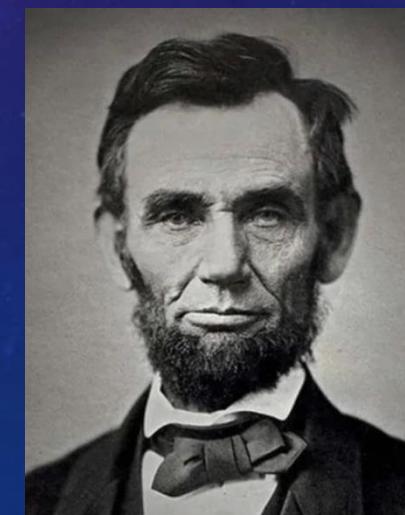
Global Variable is
Shadowed by Local Variable

```
> x = 'outer scope';
(function(){
  var x = 'inner scope';
  console.log(x);
}())
inner scope
```

Global Variable is
Shadowed by Local Variable

```
> x = 'outer scope';
(function(){
  let x = 'inner scope';
  console.log(x);
}())
inner scope
```

Global Variable is
Shadowed by Local Variable



**Initialize your
variables before
you use them.**

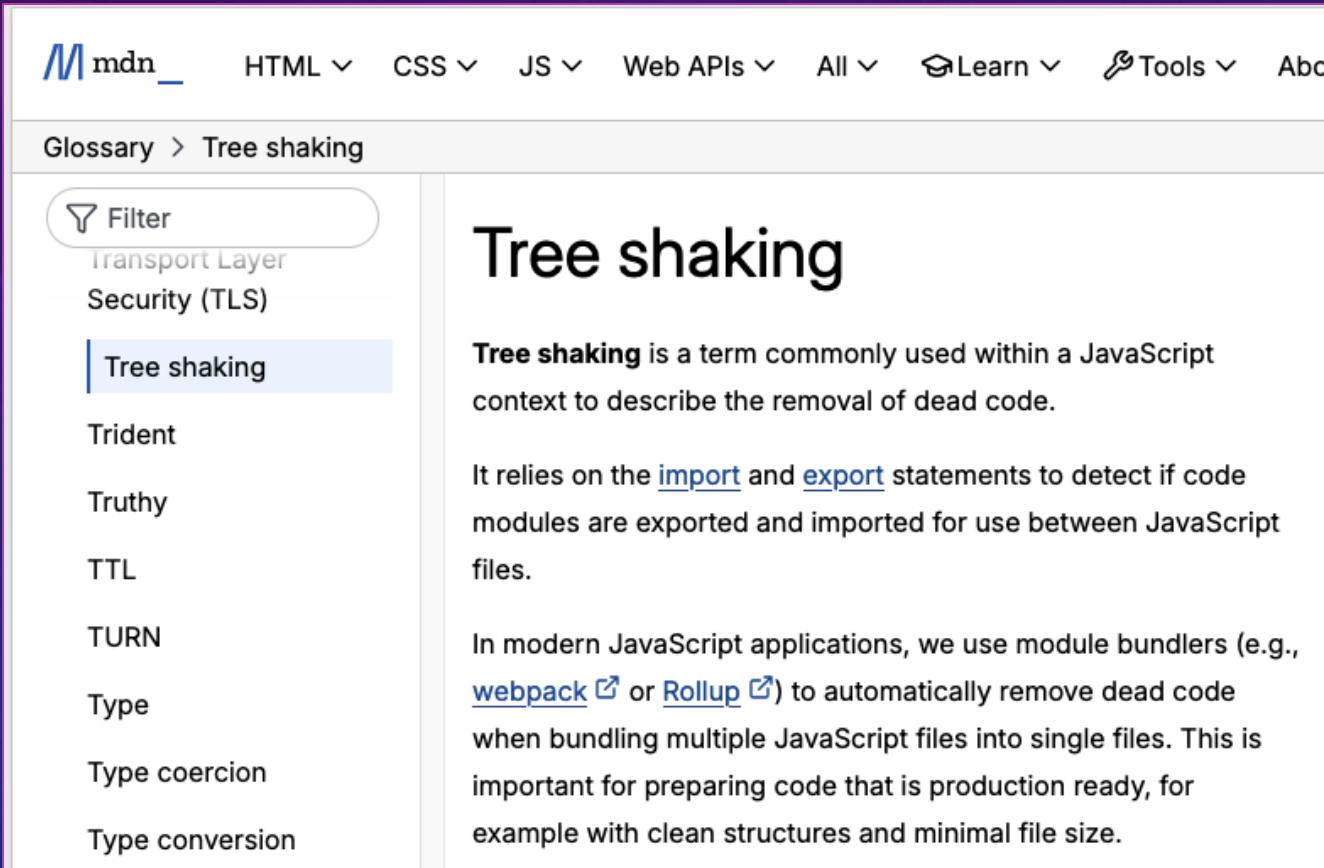
– Abraham Lincoln

TAKEAWAY

Declare, define, *then* use.

Hoisting.
Temporal Dead Zone.
Tree Shaking.
Beautiful Soup.
List Comprehension.
LoDash.

◀ Devs like to be obscure.



The screenshot shows a section of the MDN Web Docs website. The top navigation bar includes links for mdn, HTML, CSS, JS, Web APIs, All, Learn, Tools, and ABC. Below the navigation, a breadcrumb trail shows 'Glossary > Tree shaking'. A sidebar on the left contains a 'Filter' button and a list of terms: Transport Layer Security (TLS), Tree shaking (which is highlighted in blue), Trident,Truthy, TTL, TURN, Type, Type coercion, and Type conversion. The main content area has a large heading 'Tree shaking'. The text defines 'Tree shaking' as a term for removing dead code in JavaScript, explaining it relies on `import` and `export` statements. It also notes that modern applications use module bundlers like webpack or Rollup to automatically remove dead code when bundling multiple files into single files, which is important for production ready code.

Tree shaking

Tree shaking is a term commonly used within a JavaScript context to describe the removal of dead code.

It relies on the `import` and `export` statements to detect if code modules are exported and imported for use between JavaScript files.

In modern JavaScript applications, we use module bundlers (e.g., [webpack](#) or [Rollup](#)) to automatically remove dead code when bundling multiple JavaScript files into single files. This is important for preparing code that is production ready, for example with clean structures and minimal file size.