FINDING SECRETS WITH THE JAVASCRIPT CONSOLE

WITH BB KING

CONSULTING PENTESTER, REPORT WRITER, INSTRUCTOR

@BBhacKing

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

BLACK HILLS
Information Security

MODERN WEBAPP PENTESTING I AND II

REPORTING FOR PENTESTERS

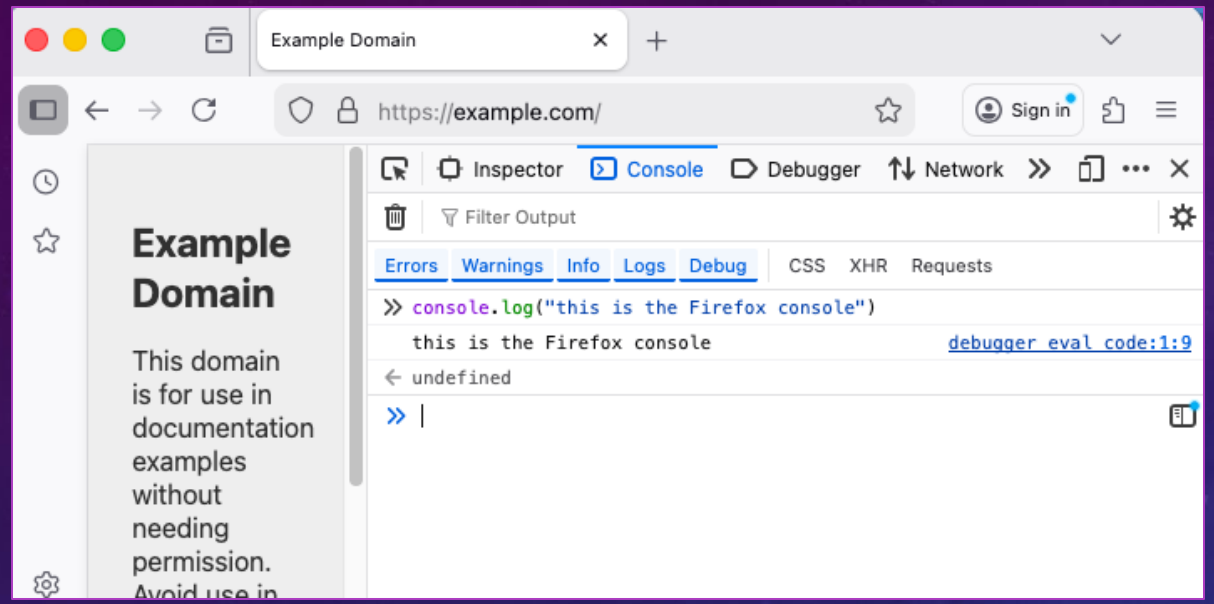
BURP SUITE WORKSHOP

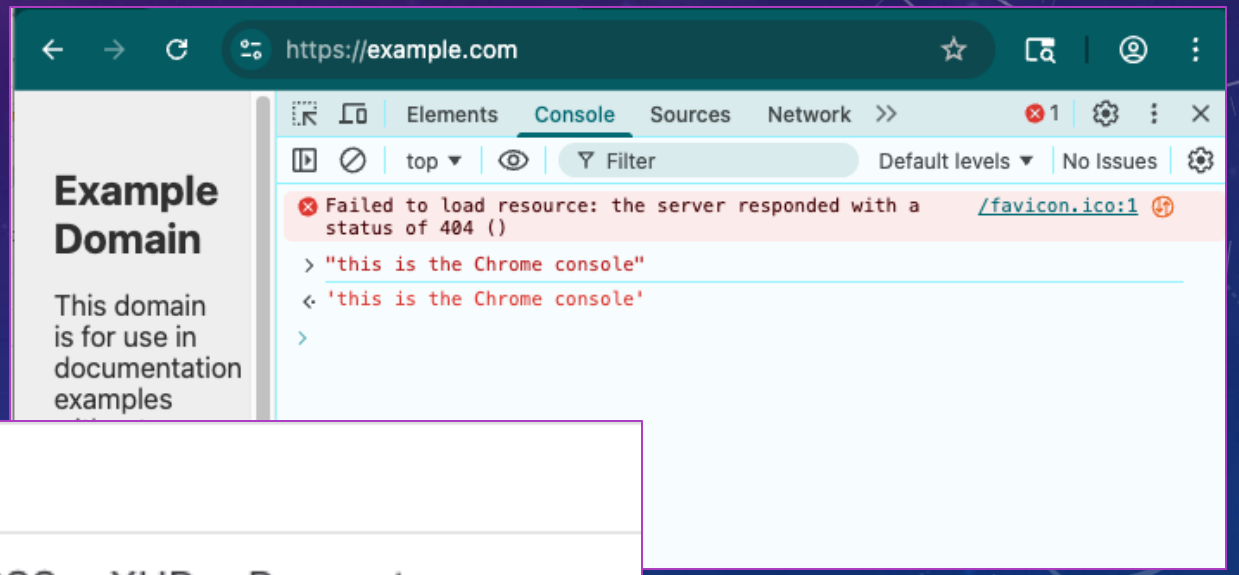
- In-Person, Simulcast, and On Demand
- MWAP I at WWHF Mile High, February 2026
- https://wildwesthackinfest.com/

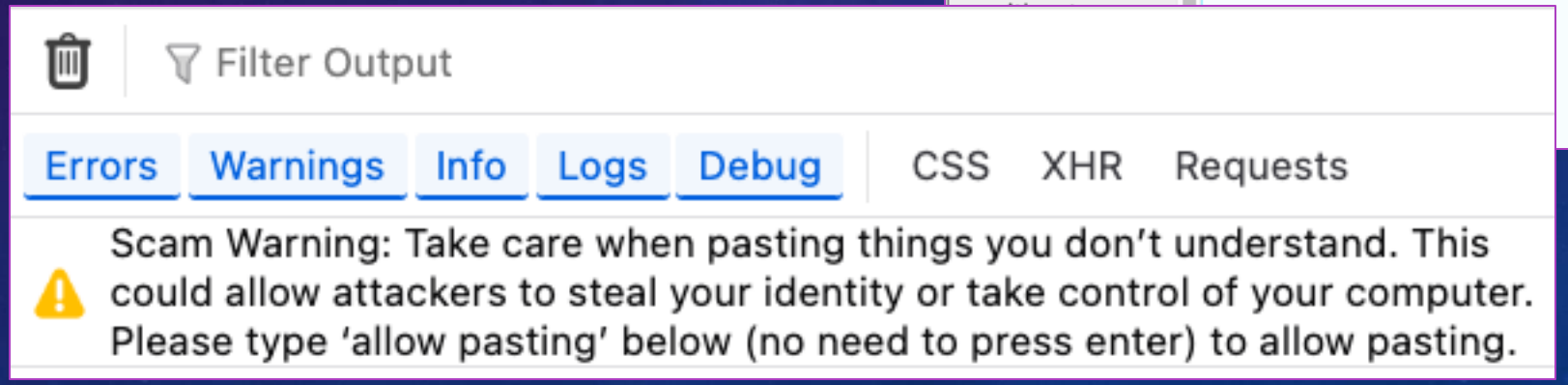
PLEASE PLAY ALONG

- 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

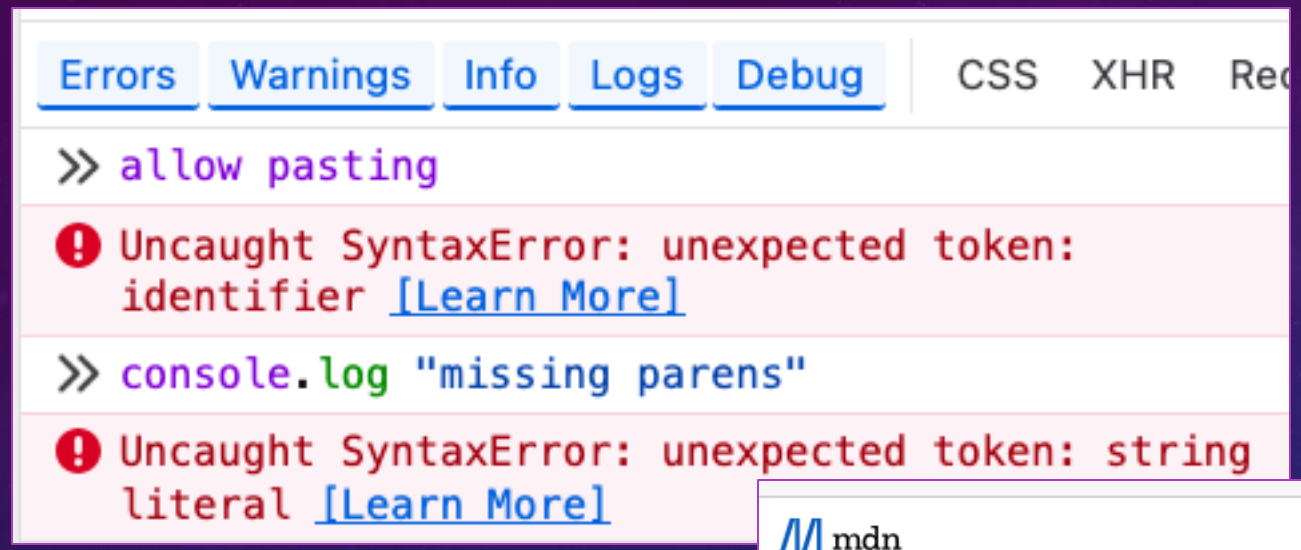






BLACK HILLS
Information Security

DEVELOPER TOOLS / CONSOLE / F12



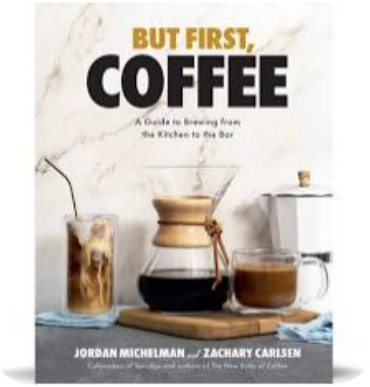
Clicking on [Learn More] may encourage learning.

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;
```

https://web.dev/learn/javascript/introduction#statements

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 coercion?)
===	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

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

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

```
» "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 ===

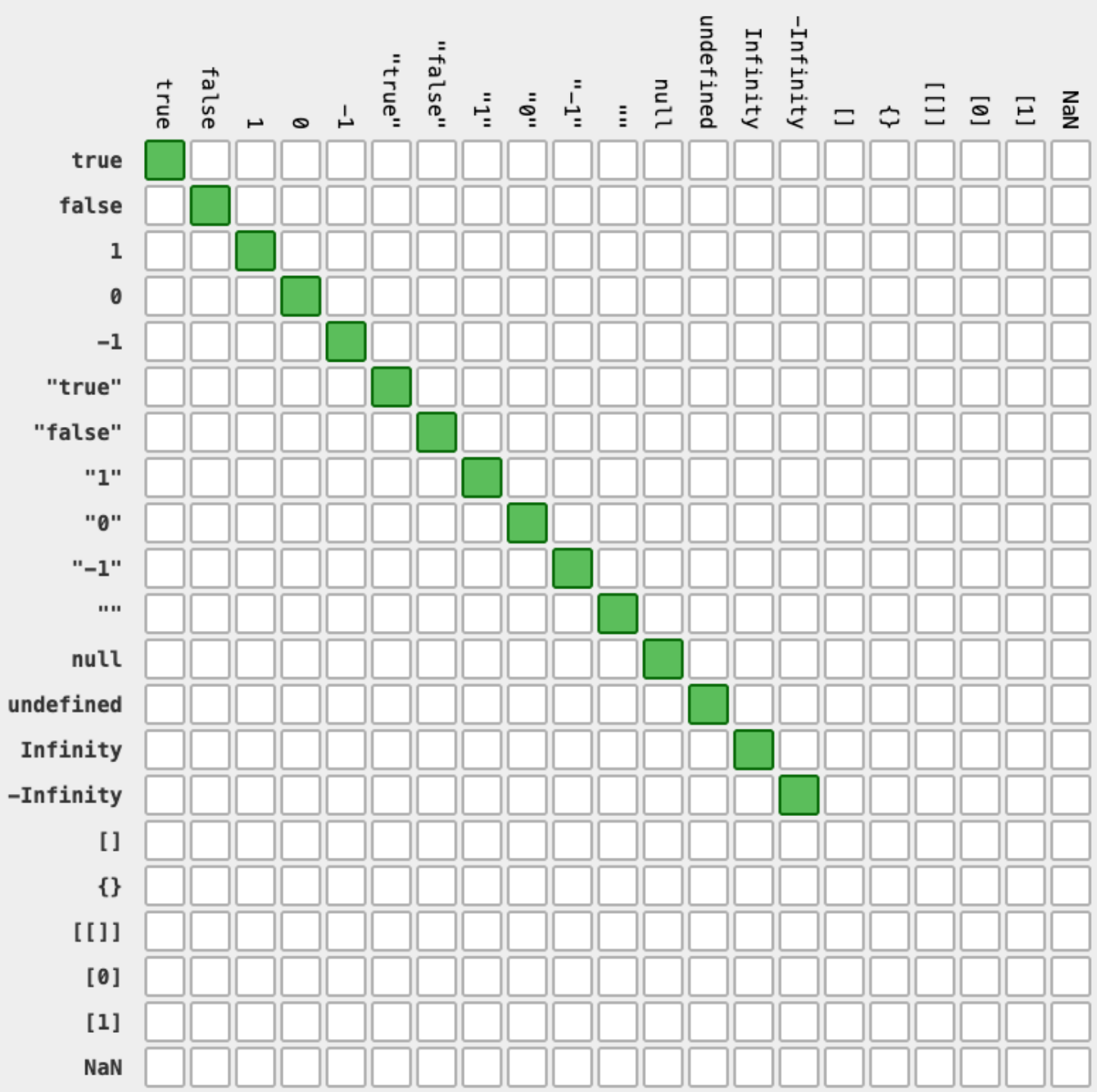
“as found”

(no coercion)

a === b

“are these the same item?”

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



TRUTHINESS TYPE B
== LOOSE EQUALITY ==

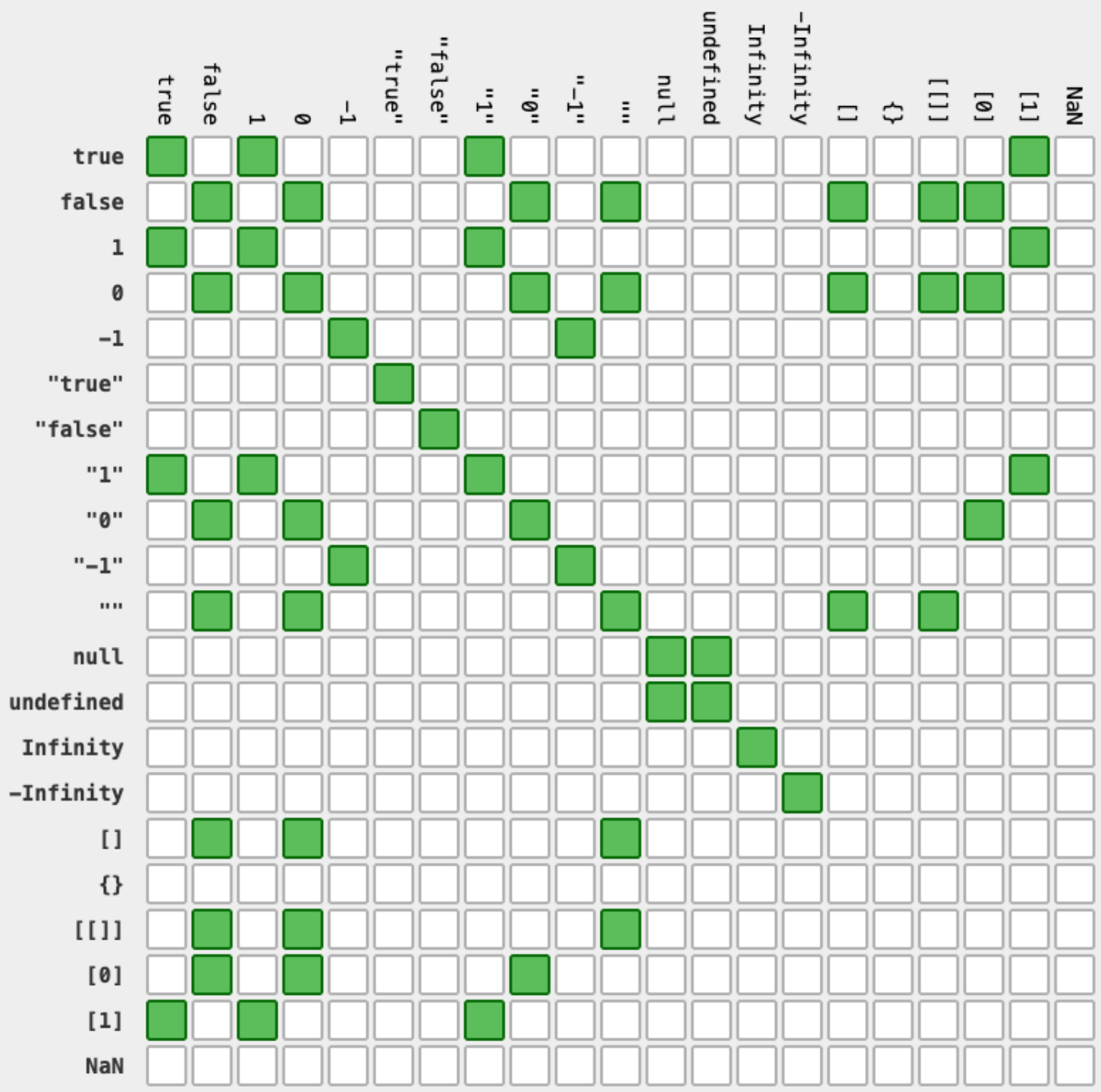
“as interpreted”

(with coercion)

a == b

“do these have the same value?”

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



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']
```

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

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

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

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

https://www.etsy.com/listing/1676528628

BLACK HILLS
Information Security

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

BLACK HILLS Information Security

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 ){}`

```
let str = "";

for (let i = 0; i < 9; i++) {
  str += i;
}

console.log(str);
// Expected output: "012345678"
```

Canonical Use Case…

…Gets Clunky With Arrays.

```
const array = ["a", "b", "c"];

for (let i = 0; i < array.length; i++) {
  console.log( array[i] )
}

// Expected output: "a"
// Expected output: "b"
// Expected output: "c"
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for#try_it

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

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for...of#try_it

BLACK HILLS Information Security

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);
```

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 ){}`

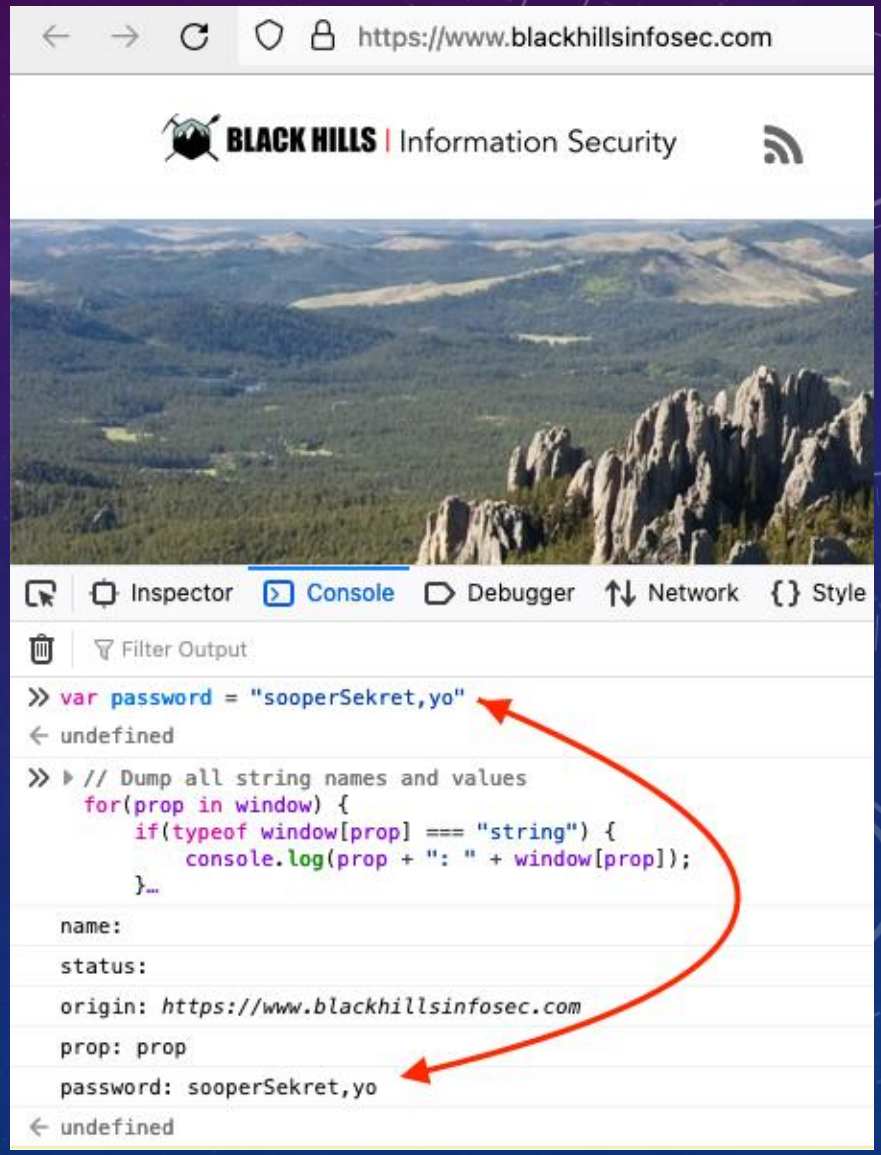
```
1  const object = { a: 1, b: 2, c: 3 };
2
3  for (const property in object) {
4    console.log(`${property}: ${object[property]}`);
5  }
6
7  // Expected output:
8  // "a: 1"
9  // "b: 2"
10 // "c: 3"
11
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for...in#try_it

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
```



BLACK HILLS
Information Security

READING FROM SENSITIVE AREAS: COOKIES

```
>> document.cookie
← "GeoIP=US:TX:[redacted]; 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

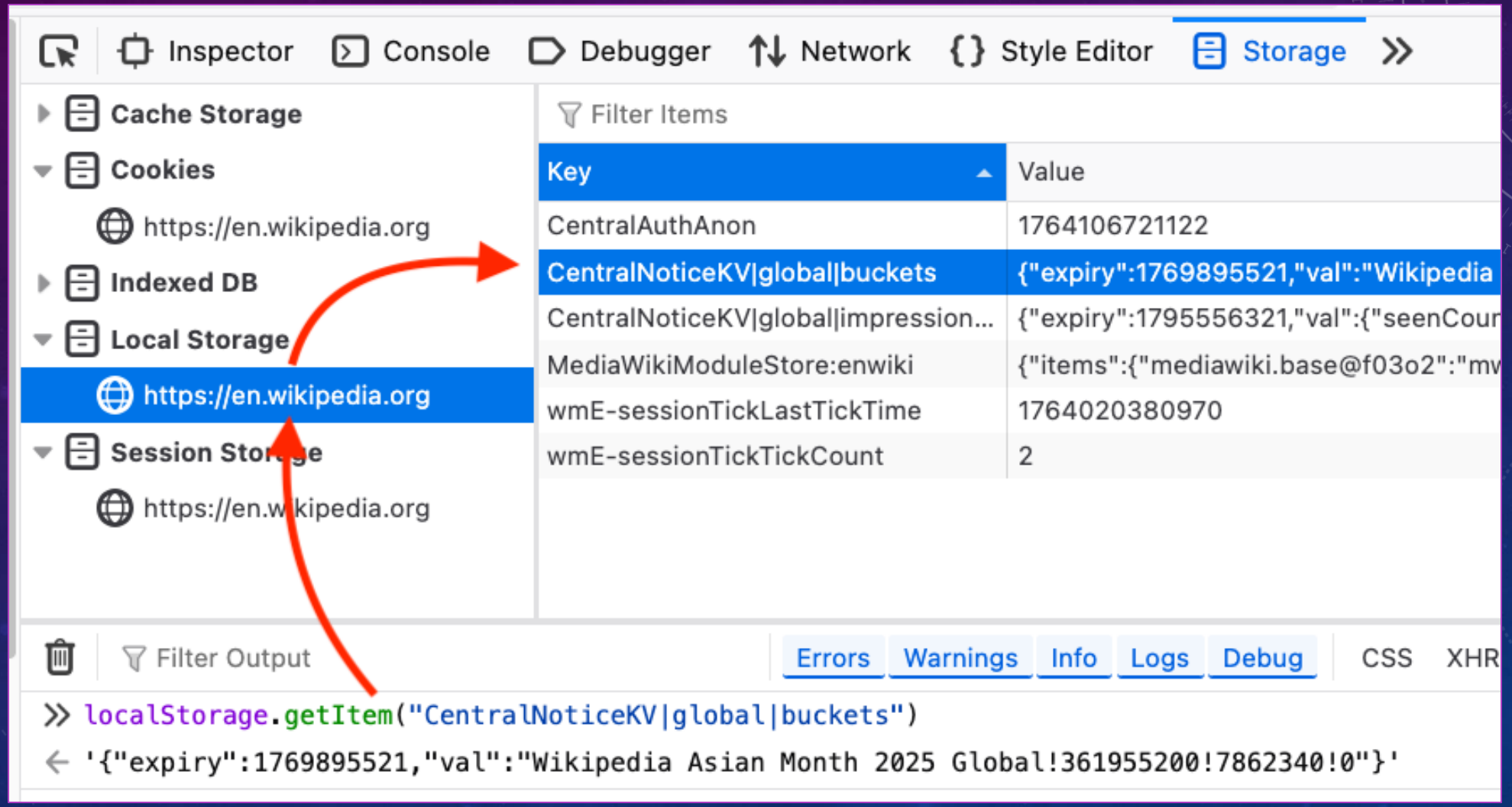
```
>> document.cookie
← "GeoIP=US:TX:[redacted]; NetworkProbeLimit=0.001;
  VEECid=d4c06d8faa991578dcb8; enwikimwuser-sessionId=a00b6b0fb70494525068"
```

This cookie has four cookies.

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

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

READING FROM SENSITIVE AREAS: DOM STORAGE



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.

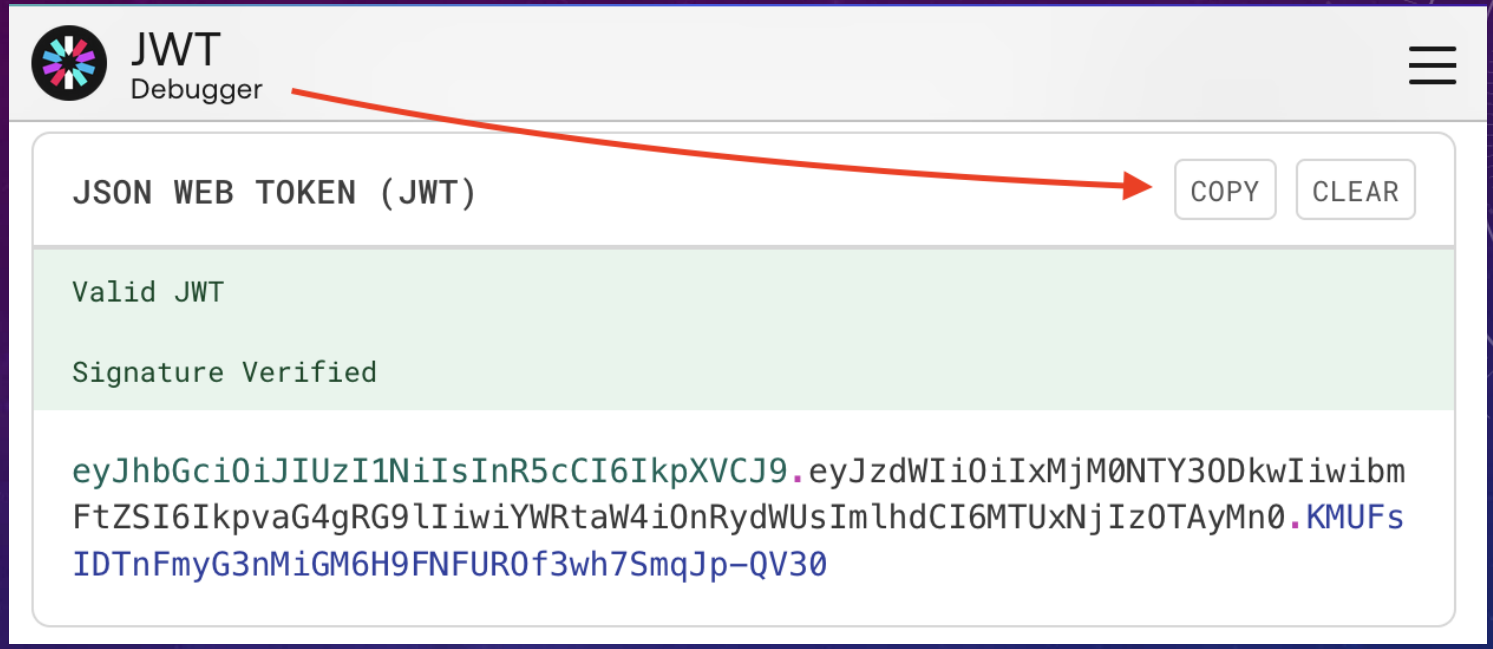
`JSON.parse()` from the last slide.

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');
```

BLACK HILLS
Information Security

CHAINING FUNCTIONS: EXAMINE A JWT IN LOCAL STORAGE

```
>> localStorage.getItem('jwt')
← "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFt
  RG9lIiwiYWRtaW4iOnRydWUsImlhdCI6MTUxNjIzOTAyMn0.KMUFsIDTnFmyG3nMiGM6H9
```

```
>> localStorage.getItem('jwt').split('.')
← ▸ Array(3) [ "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9",
```

```
>> localStorage.getItem('jwt').split('.')[1]
← "eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4
```

```
>> 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,
```

```
>> JSON.parse(atob(localStorage.getItem('jwt').split('.')[1])).name
← "John Doe"
```

BLACK HILLS
Information Security

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

BLACK HILLS Information Security

ANTISYPHON TRAINING

POWERED BY BHIS

WILD WEST HACKIN' FEST @Mile High

BB KING

MODERN WEBAPP PENTESTING

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?

BLACK HILLS
Information Security

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
```

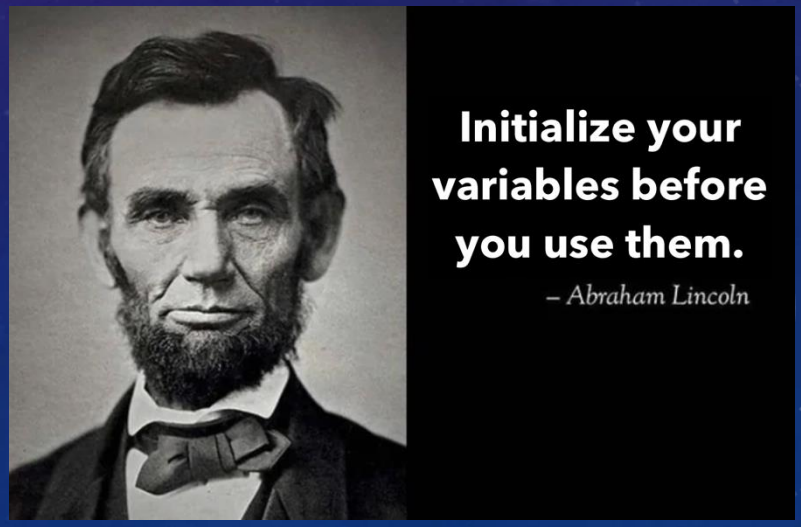
Global Variable is
Shadowed by Local Variable

```
> 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



BLACK HILLS
Information Security

TAKEAWAY

Declare, define, *then* use.

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

Devs like to be obscure.