

Debug



Agenda

- ❖ Motivation
- ❖ Debug principles
 - Validate all values of the expression
 - Isolate the problem
 - Use a debugger
 - Debug line by line
 - Use break points
- ❖ Decipher errors and exceptions
- ❖ Track back the source of the problem
- ❖ Duck debugging

Disclaimer

This workshop is **NOT** about making your code work!

It is about acquiring tools for debugging your code.
Finding ways to understand why your code doesn't work.

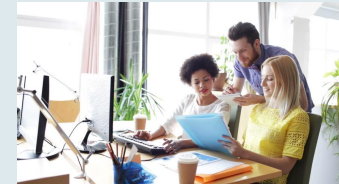
Debugging Motivation

Right now you have a lot of support!

You have teachers

TA's

each other



They can help you solve your problems.

But most of the time

You are on your own



and you will need to solve your own problems.

Debugging Motivation

Now you are learning
It is ok to ask for help.

But when you are working (or in the internship)
You will be expected to debug your own code.

Debugging Motivation

Sometimes you are almost done but suddenly
This small bug



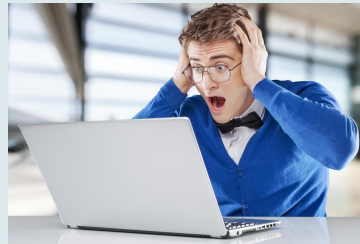
Is all that is standing between you and completing your
Exercise / task / project.

Debugging Motivation

Solving the bug might take
10 minutes



1 hour



1 day



Good debug skills will make you an efficient programmer!

Debugging Motivation

Debugging improves your coding skills

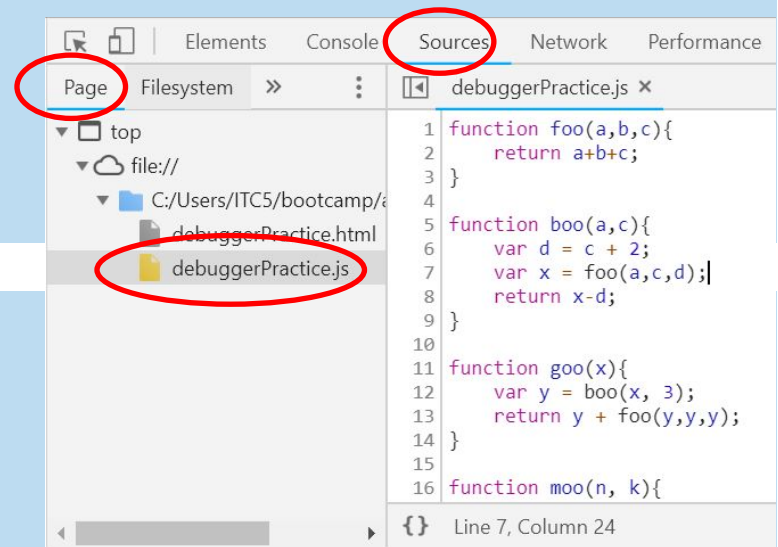
- 1 You will gain more control over the code flow
- 2 You will learn nuances of the language and how to avoid it
- 3 You will learn how to avoid bugs in the future

Chrome Devtools Debugger

Our First Tool: Debugger

1 Open the chrome browser

2 Press f12
(mac: option+cmd+J,)
Or Ctrl+Shift+I
Or inspect element



3 Sources => page => js file
Navigate to the sources tab and then to the page sub-tab,
look up you js file and open it with a click.

Debugger

Our First Tool: Debugger



Debugging browser keyboard shortcuts



F10 run the current line and then stop on the next



F11 will go into the function on the current line (don't worry about it if you don't know what a function is yet)



F8 run the code to the end, unless you hit other break points.

Debugger

Our Method: Debug line by line



Meaning: we will execute our code line by line.



In this course we will learn to use the chrome browser debugger. But, all the debuggers are very similar, once you learn how to use them.



In order for us to start, we need to add an initial break point. We will add it to the first line of the code that is executed.



To move to the next line we hit F10

Debugger

Step Into a Method



If we run into a method hitting F10 will execute all the method at once



In order to execute the method line by line we need to "step into" the method



Step into: F11




Let's debug again!

Use break points

Another method is to use **Break Points**.

How to use break points?

1. Mark a code line with a break point (click on the line number, left side)
2. Run the code from one break point to the other using:
 - F8 - run until next break point/resume:
3. Find the bug



```
Sources  Network  Performance
debuggerPractice.js x
1 function foo(a,b,c){
2   return a+b+c;
3 }
4
5 function boo(a,c){
6   var d = c + 2;
7   var x = foo(a,c,d);
8   return x-d;
9 }
10
11 function goo(x){
12   var y = boo(x, 3);
13   return y + foo(y,y,y);
14 }
```

Practice

Copy the [following code](#) to a js file that is linked to an html file.
Debug the code in the devtools and answer the questions



How many times foo is being called?

6



What is the value of b in the 3rd time foo is called?

1424



Who is calling too the second time?

zoo



What is the value of zoo(r) when we calculate variable w?

64

Practice

More Questions:



what is the value of zoo(d) the first time koo is called?

68



is there any function that is not called at all?

moo



Who is calling foo the first time it is being called?

boo that is called by goo



what is the greatest value being sent as a parameter to zoo?

34

Debugging principles

1. Isolate the problem



This is the most important rule.

Our code contains hundreds of lines, some times millions
If we were to look at each line it would take ages.
We need to know where things got wrong.



Specifically we will try to find first the line of code that raised
the error, and then the specific expression.




There are several techniques to achieve it.

Bug Scope

For practicing debugging we will have account example.

```
var accounts = {  
  a: 250,  
  b: 50,  
  c: 200  
};
```



Account will be simply a sum of money

```
function getAccount(accountName) {}
```

Returns the relevant account

```
function transfer(from, to, amount) {}
```

Move money from one account to another

```
function deposit(accountName, amount) {}
```

Add money to an account

```
transfer("a", "b", 50);  
deposit("b", 100);
```

Actions, deposit and withdraw.
For example, after the actions all accounts will have 200.

Debugging exercise

Bug 1



Check out [this code!](#)



Can you tell what is the bug?

You have 5 minutes.



Question

What type of debug methods did you use?

How did you try to solve the problem?

Debugging exercise

Bug 2



Check out [this code!](#)



Can you tell what is the bug?

You have 5 minutes.



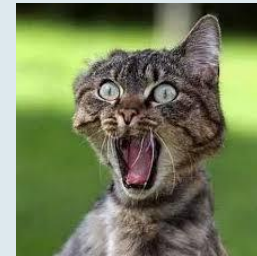
Validate all the Values in the Expression

Once upon a time . . .

there was no chrome devtool



There was no chrome browser



So what did Developers do? How did they debug?

They used the oldest method of all

Print statements,

or in js: **console.log**



Hallelujah

Print statements

WE use `console.log` to



Validate variable values



Validate result of a calculation or a function return value



Make sure something happened



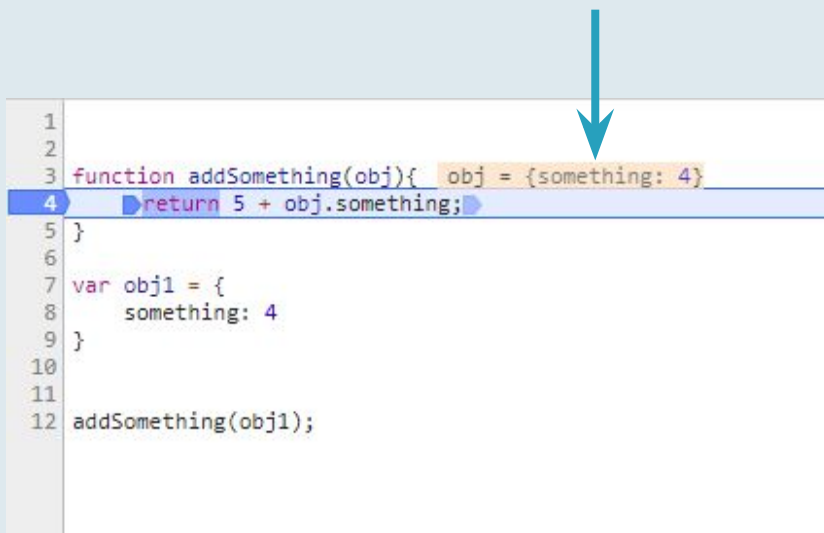
Know which steps occurred

Validate all the Values in the Expression

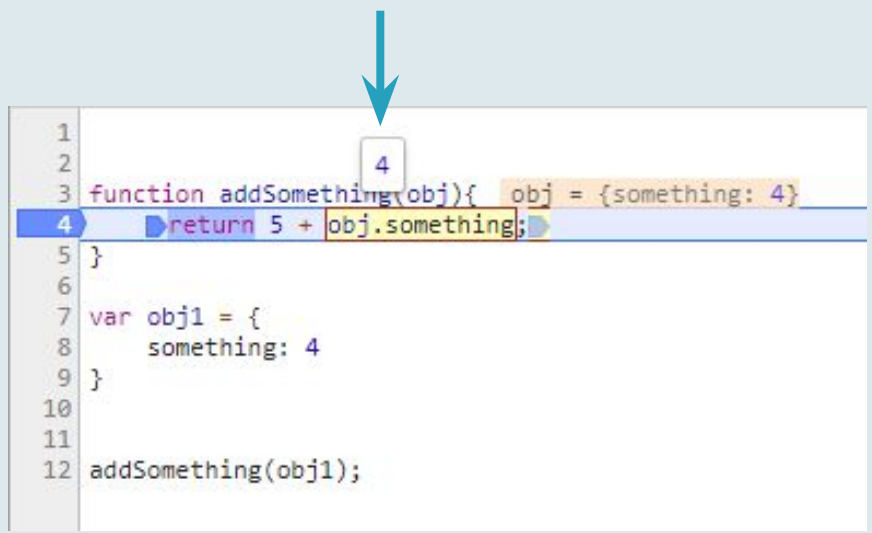
We want to be sure that our variables contain what we think they contain.

When ever we examine an expression (can be a function, loop or a code line) with the debugger we want to validate the values of the variables.

The debugger list the variables value, and present them on hover



```
1  
2  
3 function addSomething(obj){ obj = {something: 4}  
4   return 5 + obj.something;  
5 }  
6  
7 var obj1 = {  
8   something: 4  
9 }  
10  
11  
12 addSomething(obj1);
```

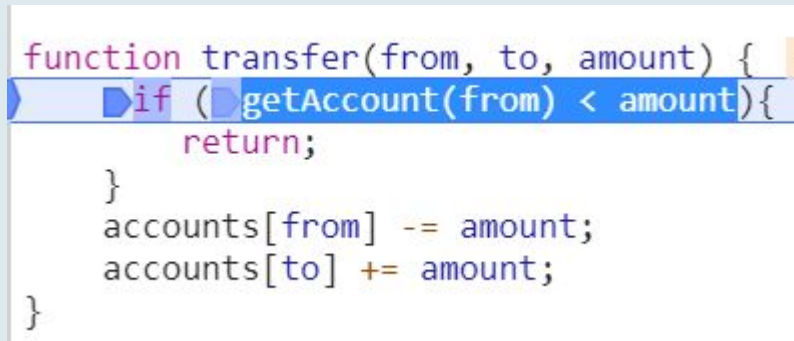


```
1  
2  
3 function addSomething(obj){ obj = {something: 4}  
4   return 5 + obj.something;  
5 }  
6  
7 var obj1 = {  
8   something: 4  
9 }  
10  
11  
12 addSomething(obj1);
```

Validate all the Values in the Expression

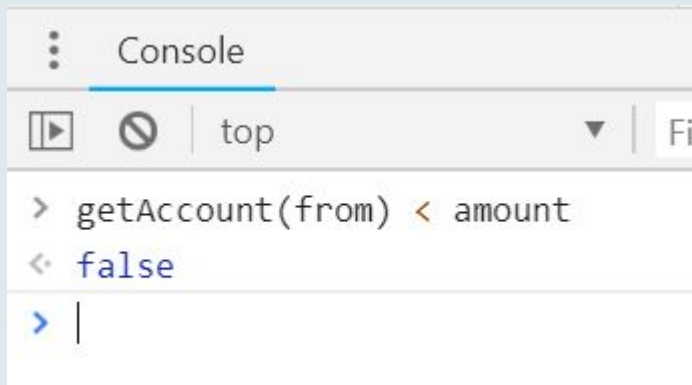
To print an expression in the console we need to:

1. Select the expression



```
function transfer(from, to, amount) {  
  if (getAccount(from) < amount){  
    return;  
  }  
  accounts[from] -= amount;  
  accounts[to] += amount;  
}
```

2. Press Ctrl + Shift + E



```
Console  
▶ | top | Fi  
> getAccount(from) < amount  
< false  
> |
```


Debugging exercise

Bug 2



Check out [this code!](#)



Can you tell what is the bug?

You have 5 minutes.



Track back the source

After we found out that there was a wrong value in one of our variables we need to understand where and how did this variable get the wrong value.

We want to trace our steps backwards



and find where it got broken.

Debug technique

Start From Scratch

Sometimes the problem is right in front of us, but we just can't see it.
That is why we do it all over (like restart).

So If we have a problem with a line of code,

We can just delete it and write it from scratch.

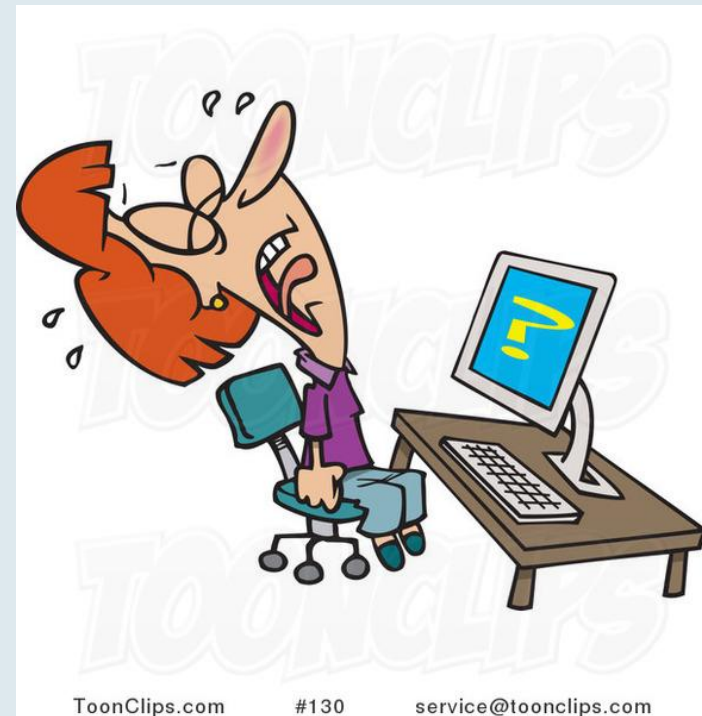
Try it!

It Happens to Everybody

When you see an error
sometimes all we can do
Is to get frustrated and think

It is NOT WORKING!

Usually it will be after long hours
in front of the computer,
Or at a very late hour,
Or a very early one.



Decipher errors and exceptions

First – try to calm down



Instead of being concentrated on the fact it is not working

**Let's try to understand
what is not working.**

**We want to be able to read errors
and understand what happened.**

Decipher errors and exceptions

Optional Error Messages

- **TypeError:**
 - Cannot read property 'name' of undefined
 - student.fixBug is not a function
- **ReferenceError:** num is not defined
- **SyntaxError:** Unexpected token :

TypeError

TypeError - Cannot read property 'name' of undefined

Meaning: you are trying to read a property from an undefined Object

Can you make an example?

Example:

```
var obj = undefined;  
console.log(obj.name);
```

TypeError

TypeError: student.fixBug is not a function

Meaning: you are trying to invoke a function that does not belong to the Object.

Option 1: the object you have is not the one you intended to have.

Option 2: something went wrong when assigning the function to the object.

Can you make an example?

Example:

```
var student = {};  
student.fixBug();
```


ReferenceError

ReferenceError: num is not defined

Meaning: you are trying to use an undefined variable

Can you make an example?

Example:

```
function add(a, b){  
  return a + b;  
}  
add(2, notDefined);
```

Or simply:

```
2 + notDefined;
```

SyntaxError

SyntaxError: Unexpected token }

Meaning: you made a syntax mistake.

Can you make an example?

Example:

```
function add (a,b){  
    return a+b;  
}  
}
```

Duck Debugging

Can I help?



Just ask the Duck

Sometimes... just telling someone about our problem, helps us to reach a solution by ourselves.

This can be anyone, a co-worker, a friend, or just a rubber duck 😊



Cheat Sheet

- ❖ Isolate the problem
- ❖ Debug line by line
 - step into: F11
 - step over: F10
- ❖ Use break points
 - run until next break point/resume: F8
- ❖ Validate all values of the expression
 - use `console.log`
- ❖ Decipher errors and exceptions
- ❖ Track back the source of the problem
- ❖ Start from Scratch