```
Springboard
                                                                Debugging + Errors
       Debugging + Errors
                                                                Download Demo Code
            « Back to Homepage
                                                                Goals
Goals
  Goals
                                                                • Catch errors using try/catch/finally blocks
Debugging
  Debugging
                                                                • Throw errors with specific messages
  SyntaxError
  SyntaxError
  ReferenceError
                                                                Debugging
  ReferenceError
  TypeError
                                                                You are going to make mistakes!
  TypeError

    Let's examine ways to better debug

  Two Kinds of Bugs
  A process for debugging

    Let's first examine some common errors

  console.log
                                                                SyntaxError
  console.log
JavaScript Debugger
                                                                • You've seen this one before!
  JavaScript Debugger
                                                                • You have to fix these right away!
  Starting Debugger
  Step Over
                                                                 "awesome
  Step Into
  Step Out
                                                                 function first( {}
  Call Stack
                                                                 let = "nice!"
  Scope
Tips To Avoid Bugs
                                                                ReferenceError
  Plan First
  Keep It Simple
  Common JavaScript Bugs
                                                                • This does not mean undefined
  Good News
Error Handling
                                                                 function sayHi(){
                                                                    let greeting = "hi!";
  Error Handling
  Let's "try" it out!
  Introducing try / catch
                                                                 sayHi();
  Another example
  Using try / catch
                                                                 greeting;
                                                                                  // ReferenceError
  When to use try/catch
  Making your own errors!
                                                                TypeError
  Exploring with try/catch
  Throwing better errors
  When should you use this?

    Accessing properties on undefined or null

  When should you not use this?
  finally
  Recap
                                                                 "awesome".splice() // TypeError
                                                                 let obj = {}
                                                                 obj.firstName.moreInfo // TypeError
                                                                Two Kinds of Bugs
                                                                • An error is thrown — easier
                                                                • You didn't get what you wanted — harder!
                                                                A process for debugging

    Make assumptions

    Test assumptions

    Prove assumptions

                                                                Repeat
                                                                console.log

    Be mindful about what you print out

                                                                • Great for a sanity check
                                                                • Even better when you add parameters
                                                                 console.log("We made it!");
                                                               JavaScript Debugger

    Watch execution of code and examine at any point

    Built into Chrome (other browsers have similiar abilities)

    Can debug in-browser code or Node

                                                                Starting Debugger
                                                               View code or adding "breakpoints":
                                                                View → Developer → Developer Tools → Sources tab
                                                                Can put breakpoint into code itself:
                                                                 function myFunction() {
                                                                    let x = 1;
                                                                     // rest of code follows ...
                                                                Step Over
                                                                         Developer Tools - https://googlechrome.github.io/devtools-samples/debug-js/...
                                                                                 Elements Console Sources Network Performance >>
                                                                                             get-started.js ×
                                                                         Sources >>
                                                                                                  * WITHOUT WAKKANTIES OK CONDITIONS OF ANT KIND,
                                                                         ▼ 🔲 top
                                                                                                  12 * See the License for the specific language gover
                                                                                                 13 * limitations under the License. */
                                                                         ▼ △ googlechrome.github.io
                                                                           ▼ devtools-samples/debug- if (inputsAreEmpty())
                                                                                                        label.textContent = 'Error: one or both inputs
                                                                              get-started
                                                                                                       return;
                                                                                                     updateLabel();
                                                                                                 {} Line 15, Column 3
                                                                        Scope Watch
                                                                                                        ▼ Local
                                                                                                         ▶ this: button
                                                                        ▼ Call Stack
                                                                                                        ▶ Global
                                                                        onClick
                                                                                          get-started.js:15
                                                                        ▼ Breakpoints
                                                                         get-started.js:15
                                                                         if (inputsAreEmpty()) {
                                                                         get-started.js:35
                                                                Run current line, but don't debug into any function calls
                                                               Step Into
                                                                        Developer Tools - https://googlechrome.github.io/devtools-samples/debug-js/...
                                                                                 Elements Console Sources Network Performance >>
                                                                         Sources
                                                                                             get-started.js ×
                                                                        ▼ 🗖 top
                                                                         ▼ △ googlechrome.github.io
                                                                                                  14 function onClick() {
                                                                              get-started
                                                                                                       return;
                                                                                                 18
19
                                                                               get-started.js
                                                                                                     updateLabel();
                                                                                                 {} Line 15, Column 3
                                                                         /
                                                                                                 Async
                                                                                                         Scope Watch
                                                                                                        ▼ Local
                                                                        Paused on breakpoint
                                                                                                         ▶ this: button
                                                                        ▼ Call Stack
                                                                                                        ▶ Global
                                                                        onClick
                                                                                          get-started.js:15
                                                                        ▼ Breakpoints
                                                                         get-started.js:15
                                                                        if (inputsAreEmpty()) {
                                                                         get-started.js:35
                                                                Run current line, stepping into any function calls
                                                               Step Out
                                                                                 Elements Console
                                                                         Sources >>
                                                                         ▼ 🔲 top
                                                                         ▼ △ googlechrome.github.io
                                                                                                  14 function onClick() {
                                                                              get-started
                                                                                                       return;
                                                                               get-started.js
                                                                                                     updateLabel();
                                                                                                 {} Line 15, Column 3
                                                                         Async
                                                                                                         Scope Watch
                                                                                                        ▼ Local
                                                                        Paused on breakpoint
                                                                                                         ▶ this: button
                                                                        ▼ Call Stack
                                                                                                         ⊳ Global
                                                                        onClick
                                                                                          get-started.js:15
                                                                        ▼ Breakpoints
                                                                         get-started.js:15
                                                                         if (inputsAreEmpty()) {
                                                                         get-started.js:35
                                                                Return from this function to caller
                                                                Call Stack
                                                                       Developer Tools - https://googlechrome.github.io/devtools-samples/debug-js/get-started
                                                                       Elements Console Sources Network Performance Memory Application >>
                                                                                                               get-started.js ×
                                                                                           * WITHOUT WARRANTIES OR CONDITION
                                                                                           * See the License for the specifi 1 Paused on breakpoint
                                                                        ▼ 🔲 top
                                                                                           * limitations under the License.
                                                                        ▼  googlechrome.github 13
                                                                                            unction onClick() {
                                                                         ▼ devtools-samples/ 15
                                                                                            if (inputsAreEmpty()) {
                                                                                             label.textContent = 'Error: o
                                                                            get-started
                                                                                                                   ▼ Call Stack
                                                                                              return;
                                                                            get-started.js
                                                                                            updateLabel();
                                                                                                                    getNumber1
                                                                                         21 function inputsAreEmpty() {
22  if (getNumber1() === ''
23   return true;
                                                                                                                    onClick
                                                                                            } else {
                                                                                             return false;
                                                                                        28 function updateLabel() {
29    var addend1 = getNumber1();
30    var addend2 = getNumber2();
31    var sum = addend1 + addend2;
32    label textCentent = addend1
                                                                                            var sum = addend1 + addend2;
                                                                                        {} Line 22, Column 7
                                                                Shows stack of function calls that got you here
                                                                Scope
                                                                        Elements Console Sources Network Performance Memory
                                                                        ▼ 🔲 top
                                                                        ▼  googlechrome.github 25 26
                                                                                              return false;
                                                                                                                   ▶ Threads
                                                                         ▼ line devtools-samples/
                                                                                                                   ▶ Watch
                                                                                         28 function updateLabel() {
                                                                            get-started
                                                                                       29     var addend1 = getNumber1();
30     var addend2 = getNumber2();
31     var sum = addend1 + addend2;
32     label.textContent = addend1 +
                                                                                                                   ▼ Scope
                                                                                        34 function getNumber1() {
                                                                                            return inputs[0].value;
                                                                                         37 function getNumber2() {
                                                                                            return inputs[1].value;
                                                                                         40 var inputs = document.querySelecto
                                                                                         41 var label = document.querySelector
                                                                                        {} Line 31, Column 21
                                                                Shows current value of variables
                                                                Can click to change value
                                                                Tips To Avoid Bugs
                                                                Plan First
                                                                       -Sam Redwine, 1988
                                                                Keep It Simple
                                                               Common JavaScript Bugs
                                                                == is very loose about comparisons (=== isn't)
                                                                     • 7 == "7"
                                                                • Objects & arrays are not equal to similar objects & arrays
```



## Notice the important difference here!

} catch (err){

**Good News** 

-Edsger W. Dijkstra

Connection failures

functionThatDoesNotExist();

console.log("did we make it?");

functionThatDoesNotExist();

The API is down

Let's "try" it out!

try {

• Sometimes errors can not be avoided!

- Introducing try / catch try {
- } catch(err){
- return `Hello \${firstNameLetter}.\${lastNameLetter}`; What happens when we don't pass in an object? What happens when our object does not have the correct keys or values?

function displayInitials(user){

Using try / catch function displayInitials(user){

let firstNameLetter; let lastNameLetter;

When to use try/catch

Making your own errors!

try {

Think about using try/catch when: • The output of your program is unpredictable • There is any chance that an unexpected error may occur You don't want a tremendous amount of conditional logic

console.log("an error is coming....");

console.log("what happened?", err);

throw "What just happened?!?";

console.log("an error is coming...."); try { throw "Oh no!";

} catch (err) {

happened.

your own errors.

finally

When should you use this?

**Throwing better errors** Instead of just using the throw keyword, you can make a new Error object using the new keyword. console.log("an error is coming...."); try { throw new Error("Oh no!"); } catch (err) { console.log("what kind of error?", err.name);

Notice that we get more information here including where the error occured!

• any time you want to hide errors from a user • any time you want to produce custom errors for your users' benefit. foreseen during development

If you know an error is going to occur, because in this case you would want to debug the problem, not hide it. You still have to debug on your own!

try { undefined(); // this will throw a TypeError } catch (err){ console.log("something went wrong!", err); } finally {

You will rarely find yourself using this, but it's good to know if you want to be explicit about what to run no matter what in a try/catch block Recap • A SyntaxError happen when JavaScript is unable to parse certain syntax scopes!

 You can throw your own errors using the throw keyword • Using try/catch/finally blocks are an excellent way to handle errors when they may be unpredictable

• Define three of the most common errors in JavaScript and how to debug them • Examine the Chrome Dev Tools for debugging

🌋 Springboard

• Thrown when you try to access a variable that is not defined

 Trying to do something with a type that you can not • Invoking ("calling") something that is not a function

console.log("The value of x is --->", x);

Click left of line of code to add a blue breakpoint debugger; // <-- will always stop here</pre>

Window

Window

get-started.js:35

get-started.js:15

\* WITHOUT WARKANTIES OR CONDITIONS OF ANT KIND, E

\* See the License for the specific language gover \* limitations under the License. \*/ ▼ devtools-samples/debug- 15 if (inputsAreEmpty()) { label.textContent = 'Error: one or both inputs

> Developer Tools - https://googlechrome.github.io/devtools-samples/debug-js/... Sources Network Performance >> \* WITHOUT WARKANTIES OR CONDITIONS OF ANY KIND, E
>
> \* See the License for the specific language gover \* limitations under the License. \*/ devtools-samples/debug- 15 if (inputsAreEmpty()) { label.textContent = 'Error: one or both inputs Window

Breakpoints XHR Breakpoints ▶ DOM Breakpoints ■ Global Listeners Event Listener Breakpoints

Developer Tools - https://googlechrome.github.io/devtools-samples/debug-js/get-started Paused on breakpoint

(That's a particularly excellent intermediate book, by the way)

• [1, 2, 3] !== [1, 2, 3] • Calling function with missing arguments makes those arguments undefined • Calling function with extra arguments is ignored (the extra arguments are ignored). • Getting a missing property from object/index from array is undefined

If debugging is the process of removing bugs, then programming must be the process of putting them

• You will have lots of chances to practice this valuable skill! **Error Handling** 

• This especially happens when working with external APIs / other people's data

• Instead of errors crashing our program, let's handle them gracefully!

• Bugs are an opportunity to improve debugging skills & to learn something

console.log("something went wrong!", err); console.log("did we make it?");

// place the code you would like to try to run // if an error occurs, run whatever code we place in here! // keep going! try and catch create block scope, so if you define a variable inside using let or const it will only exist in that block. The parameter to *catch* is optional, but we highly recommend using it so you can see what the error is and any information about it! **Another example** 

let firstNameLetter = user.firstName[0].toUpperCase(); let lastNameLetter = user.lastName[0].toUpperCase();

firstNameLetter = user.firstName[0].toUpperCase();

lastNameLetter = user.lastName[0].toUpperCase(); } catch(e){ return "Invalid input!"; return `Hello \${firstNameLetter}.\${lastNameLetter}`; Strive to only place code that will throw an error inside your try/catch block!

console.log("did we make it?"); **Exploring with try/catch** 

You can actually create your own errors in JavaScript using the throw keyword

console.log("what is the message?", err.message); console.log("where did it happen?", err.stack);

All we see here is the string that we passed to throw with no information about the kind of error or where it

• it works as a nice fallback in areas of your code that could fail under unusual circumstances that were not • it can hide overly-technical error messages from users who wouldn't understand them anyhow • use it for portions of your code where you suspect errors will occur that are beyond your control When should you not use this?

We'll discuss what *new* does in much more detail later in the course, but for now you should use it when making

thrown or not console.log("we're all done!");

try/catch blocks come with another optional keyword finally which will execute code regardless if an error is

• A ReferenceError occurs when JavaScript tries to access a variable that is not aware of - remember your • A TypeError occurs when JavaScript runs an invalid expression on the wrong type