



Hello Code 2

JavaScript

Duane DuaneHaworth@gmail.com



Motivation

Fullstack.io, now <https://www.newline.co>, asked their readers “If your best friend asked you how to learn how to program, what would you tell them?” Instead of getting a list of URLs, people were giving great advice on how to **approach the problem of learning how to program**.

- Believe you can do it
- Have a project idea, what do you want to build
- What technology do you want to use
- Lots of resources
- Mentor
- Don't need a CompSci degree



Why Learn JavaScript

- Language used by browsers
- Enhance User Experience
 - Add/remove/alter HTML
 - Add/remove/alter CSS
- Get and Submit Data
- Games
 - <https://codepen.io/hellokatili/pen/xwKRmo>
- Forgiving Language
 - Pros/Cons
- Syntax rules similar to others
 - Java, C, C++, C#



What We're Going to Use in this Course

- W3Schools
 - <https://www.w3schools.com/js/>
- Glitch
 - www.glitch.com
- GitHub
 - <https://github.com/DSHaworth/HelloCode2>
- Questions



Preliminaries

Syntax

- * Language Rules

Statement

- * Instruction executed by computer
- * Executed in order they are written

Variables

- * Hold data of various data types and can be changed at anytime

Data Types

- * boolean
- * number
- * string
- * object
- * undefined
- * null



Preliminaries continued

Conditions

- * Execute block of code when condition true

Loops

- * Execute block of code while condition true

Functions

- * Block of code designed to perform a task

Reserved Words

- * Words claimed by JavaScript. Off limits

Comments

- * Not statements
- * Not Executed
- * Document Code
- * Prevent Execution
- * Single Line //
- * Multi-Line /* ... */



Adding JavaScript to a Webpage

Internal Script

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Hello!</title>
  </head>
  <body>
    <script>
      alert("JavaScript says Hi");
    </script>
  </body>
</html>
```

External Script

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Hello!</title>
  </head>
  <body>
    <script src="script.js"></script>
  </body>
</html>
```

Functions

Functions are an important concept, so we cover it early





Functions

- Block of code that performs a task
- Executed when something “invokes” it.
- Can take zero or more arguments
- Can return a value.

Right now, we’re looking at functions provided by the JavaScript Window API.

- API
 - Application Programmer Interface
- Window Object
 - Global
- Some Window API functions we’ll use now
 - alert
 - prompt

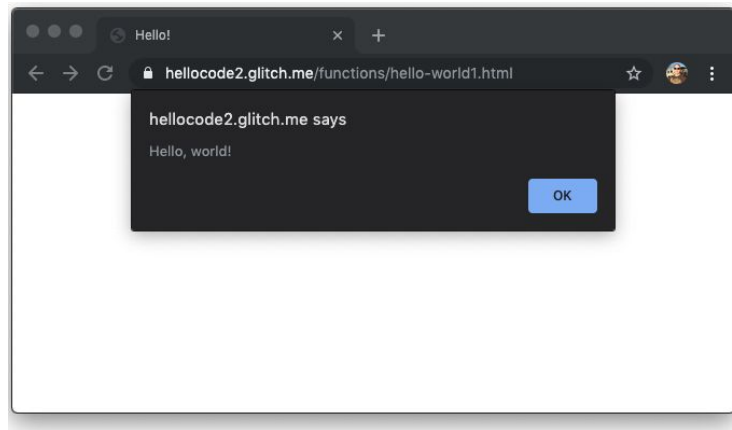
CAUTION

- Case Matters!!!!!
 - `Alert` is not the same as `alert` is not the same as `ALERT`
- This is true for `variables` and `functions`
- Keep Code Blocks Aligned



alert

- Displays an alert box.
 - The developer determines the message to display





hello-world1.html

- Hello Code 2 - Exercises
 - Hello World 1
- Button click event
- Executes function
- Displays Alert - “Hello, World”

Variables and Data Types

—



Variables and Data Types

- Variables
 - Store any Data Type
 - Can change (hence the name)
- Declare variable using `var`

```
var state = "Nebraska";  
var age = 152;  
var isState = true;  
var cities = ["Omaha", "Lincoln"];
```

- Data Types
 - string
 - number
 - boolean
 - object
 - array
 - undefined
 - null



Data Type: String

- Anything within quotes
- Type of quotes don't matter. (but it does)
 - "It's easier with double quotes"
 - 'It's easier with double quotes' (Error)
 - 'It\'s easier with double quotes'
- Rule of thumb for numbers
 - If you don't do math on it, it's a string
 - Phone number
 - Social Security Number
 - Date (though there is a date object)
- Examples
 - "Hello Code 2 Rocks"
 - 'Hello Code 2 Rocks'
 - '10/11/2019'
 - '800-555-1212'



Data Type: Number

- Any type of number
 - With decimal or no decimal
- Numbers in quotes are strings
- JavaScript will try to convert strings to numbers.
 - Safer to convert

- Examples
 - 1
 - 3.14
 - Numbers, but may not be what you expect
 - 10/11/2019
 - 800-555-1212



Data Type: Boolean

- One of two values
 - true
 - False
 - Everything with a value is true
 - Everything without a values is false
- True values
 - "Hi"
 - 3
 - "false"
 - This is a string
 - False values
 - 0
 - ""
 - null
 - undefined



Data Type: Object

- Contain many values
 - Name:Value Pairs
- Can contain **functions**

JavaScript Object Types

- Date

- Example

```
var loc = {  
  city: "Omaha",  
  state: "NE",  
  highTemp: 50,  
  lowTemp: 30,  
  statehood: "3/1/1867",  
  inUS: true  
};
```



Data Type: Array

- Store multiple values in a single variable
- Has an API

Example:

```
var cars = ["Saab", "Volvo", "BMW"];
```



Data Type: Undefined and Null

- Undefined
 - Variable declared, but no value assigned.
 - It doesn't know what it is yet
 - Number, string, boolean, etc..
- Null
 - Represents no value



Variables - Details, details, details

- Naming Rules (**syntax**)
 - Can contain letters, numbers, underscores, dollar signs
 - Must BEGIN with a letter, underscore, or dollar sign
 - ARE CASE SENSITIVE
 - Reserved words cannot be used as names
- Tips
 - Use a variable name that represents the data to be stored
 - Do: `var firstName = 'John';`
 - Don't: `var x = "7/4/1776";`

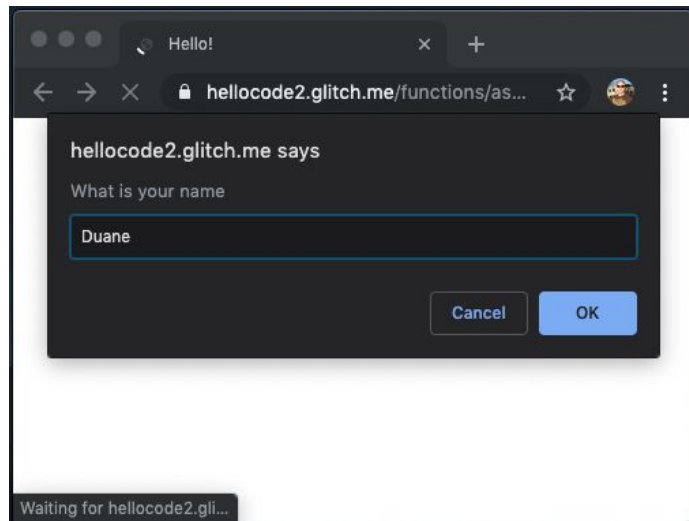


hello-world2.html

- Hello Code 2 - Exercises
 - Hello World 2

prompt

- Displays a prompt box asking for input.
 - The developer determines the message to display
- Prompt *returns* the value that was entered by the user
 - *User input*
- We assign the *return value* to a variable to capture





Concatenation

- Fancy word for putting strings together.
- In programming, there are multiple ways of doing the same thing.
- Examples
 - `var name = prompt("What is your name?");`
 - `var greetings = "Hello, " + name;`

Another way...

- `var greetings = "Hello, ".concat(name);`



ask-name.html

- Hello Code 2 - Exercises
 - Ask Name



Create your own function

- function names follow the same restrictions as variable names
 - Letters, numbers, underscore, dollar signs
 - Must start with a letter, underscore, or dollar sign
 - Can take 0 or more arguments (parameters)
 - Can, but doesn't have to, return a value
 - Tip
 - Have function name represent what it does.
 - A function does something



ask-name-function.html

- Hello Code 2 - Exercises
 - Ask Name Function




Call JavaScript from HTML

- So far, we've had to refresh our page (F5) to have JavaScript execute our code.
- Now we're going to *invoke* our JavaScript when we press an HTML button.
- The HTML button provides an **attribute** called **onclick** where we can assign our JavaScript function to execute.



ask-question.html

- Hello Code 2 - Exercises
 - Button Ask Question



askQuestion() vs prompt()

You may have noticed, we're not doing anything different in `askQuestion()` than what we can get from `prompt()`.

What's the point of `askQuestion()`?

We're going to make `askQuestion()` better than `prompt` by doing some **Data Validation**



Review

- 1) `alert("Hello, World")`
- 2) `alert(greeting)`
- 3) `alert(prompt("name"))`
- 4) `alert(prompt(question))`

The idea is to see a progression from spitting out hard-coded string, to string determined by the user



What's Next

Now we're going to look at the User Input

- 1) Did user press Cancel?
 - a) Return null
- 2) Did user press Ok without entering anything?
 - a) Return ""

We don't want our app to respond with "Hello, null!" or "Hello, !"

We want to **validate** user input



Challenge 10 Minutes

https://www.w3schools.com/js/js_htmlDOM.asp

https://www.w3schools.com/js/js_htmlDOM_methods.asp

Some liked the Age calculator. Write your own Age Calculator. :-)

DOM Object

Navigate to

[https://github.com/DSHaworth/HelloCode2](https://github.com/DSHaworth>HelloCode2)

Click on **calculate-age-begin.html**

In the function **askDob** is the *algorithm* for calculating age.

Fill in under the **Get** and **Display**

Decisions

Making decisions based on comparisons and logic





Decision Making

- Problems with our askQuestion function
 - Returns null on cancel
 - `Hello, null!` doesn't make sense
 - Returns "" when nothing entered
 - `Hello, !` doesn't make sense
- This is where things start to get intense.
- Several "moving parts" are being introduced here.
- Decisions consists of
 - Condition statements
 - Comparisons
 - Logic
 - Evaluated Left to Right



Comparison Operators

==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	great than or equal to
<=	less than or equal to

Remember

= assignment operator (assigns value to variable)



Logical Operators

- `&&` AND - Both conditions need to be true for the whole thing to be true
- `||` OR - Only one condition needs to be true for the whole thing to be true
- `!` NOT - Reverses current value

Programmer Joke:

```
!false - It's funny because it's true.
```



Truth Tables

Truth Tables AND

A	B	A && B
T	T	T
T	F	F
F	T	F
F	F	F

Truth Tables OR

A	B	A B
T	T	T
T	F	T
F	T	T
F	F	F



Decisions - Putting it all together

- Condition Statements are the test
 - `if else`
 - `switch`
- Comparison Operators are the evaluation
 - `==, ===`
 - `!=, !==`
 - `>, >=, <, <=`
- Logical Operators combine evaluations
 - `&&`
 - `||`
 - `!`



condition01.html

- Hello Code 2 - Exercises
 - Condition null



condition02.html

- Hello Code 2 - Exercises
 - Condition empty string



logic01-bad-demo.html

- Hello Code 2 - Exercises
 - Test for the null and empty string
 - Logic Bad 1



logic01.html

- Hello Code 2 - Exercises
 - Putting the null test and empty test together
 - Logic Fixed 1



String API

Common String Methods

- `toLowerCase()`
- `toUpperCase()`
- `trim()`
- `replace(oldValue, newValue)`
- `indexOf(valueToFind)`
- `lastIndexOf(valueToFind)`



logic02-bad-demo.html

- Hello Code 2 - Exercises
 - Problem....User enters spaces
 - Logic Bad 2



logic02.html

- Hello Code 2 - Exercises
 - Conditions are tested left to right
 - Logic Fixed 2



Challenge.html

- Hello Code 2 - Exercises
 - logic02-assignment.html
 - Put all the logic in askQuestion
 - If result = null or "" after trimming,
 - return null
 - return result trimmed
 - If name
 - Display name
 - Else
 - Display error

Review

Functions, Comparing, and Logic





Functions

Functions are commands.

You can send arguments (within parenthesis)

You can return a value

```
function multiplyTwoNumbers (num1, num2) {  
    return num1 * num2;  
}
```

```
var result = multiplyTwoNumbers (5, 6);
```

API Application Programmer's Interface

```
string.trim()  
document.getElementById ("id")  
Date.parse ("12/25/2019")
```

User Defined

```
multiplyTwoNumbers (3, 6)
```



Comparison and Logical Operators

Comparison Operators

- == - Equal by value only
- === - Equal by value and by type
- != - Not equal by value only
- !== - Not equal by value and type
- > - Greater than
- >= - Greater than or equal to
- < - Less than
- <= - Less than or equal to

Logical Operators

- && - AND - Both sides must be true to be true
- || - OR - If either side is true, it's all true
- ! - Reverse



Conditional Statements (if)

Comparison Operators

- == - Equal by value only
- === - Equal by value and by type
- != - Not equal by value only
- !== - Not equal by value and type
- > - Greater than
- >= - Greater than or equal to
- < - Less than
- <= - Less than or equal to

Logical Operators

- && - AND - Both sides must be true to be true
- || - OR - If either side is true, it's all true
- ! - Reverse

Loops

Interacting with HTML Input Elements



Loops???

- When working with data, you're inevitably going to be working with arrays.
 - An Array is a series of data, strings, numbers, dates, and/or objects that contain any combination.
 - `[1, 2, 3]`
 - `["Peter", "Paul", "Mary"]`
 - `["1/1/2019", "11/11/2019", "12/25/2019"]`
 - `HTML Elements`
- Loops are used to *iterate* through an array
 - When looping through an Array, it always starts at 0
 - Array is an object, it has properties and methods
 - `.length`
 - `.sort()`



Demo Array

- Go to **Developer Tools** on your browser
 - Go to **Console**
- Enter:
 - `var names = ["Peter", "Paul", "Mary"]`
 - `names[0]`
 - `names[2]`
 - `names[3]`
 - `names.length`
 - `names.sort()`



Loops???? Continued

- Loops need to know:
 - Where to start
 - Where to stop
 - test (CONDITIONS)!!!!
 - How to move on
- Three Types of Loops
 - for
 - while
 - do while



Loops **CAUTION**

- Loops without a stopping point are called **ENDLESS LOOPS**.
 - **Endless Loops** are BAD
 - Can bog down browser to the point of having to force browser to close
- It's going to happen.
 - It happened to me writing the demos for this.
 - I changed variable names to something more clear, but didn't change them all.



for loop

- Ugliest of the loops
 - Most common and most useful
 - Everything is on one line (where to start, stop, and how to move on)

```
for(start ; stop test ; move on){  
    Everything between the {} is in the loop.  
    BEST PRACTICE: ALWAYS USE {}  
}
```



for loop

```
var start = 0;  
var stop = 10
```

```
for(idx = start ; idx < stop ; idx++){  
  console.log(idx);  
}
```

idx++?????????

- ++ is an incrementer
 - Shortcut for `idx = idx + 1;`



for loop challenge 01 10 minutes

- 03-loops
 - For-loop-challenge01-begin.html

```
for("start" ; "stop test" ; "move on")
```

- Replace “start”; “stop test”; “move on” with valid JavaScript
 - See previous slide
 - Use provided variables

“But Teach, you didn’t define `idx` in previous slide!!!”

Hoisting - JavaScript will create a definition for you.

"use strict" forces you to define all variables... Makes JavaScript more “Type A”



for loop challenge 01 final

- 03-loops
 - For-loop-challenge01-begin.html

```
for("start" ; "stop test" ; "move on")
```

- Replace “start”; “stop test”; “move on” with valid JavaScript
 - See previous slide
 - Use provided variables

“But Teach, you didn’t define `idx` in previous slide!!!”

Hoisting - JavaScript will create a definition for you.

"use strict" forces you to define all variables... Makes JavaScript more “Type A”



for loop challenge 02 10 minutes

- Use for-loop to count by 2
 - Change the “Move On”
 - Increment by one
 - `idx++; //ADD 1 to idx`
 - `Idx += 1; //Add 1 to idx`
 - Equivalent to `idx = idx + 1`
 - Count by 2
 - `idx += 2; // Add two to idx`
 - Equivalent to `idx = idx + 2;`



demo-for-loop-change-colors.html

- Create boxes
 - Turn boxes blue and red



demo-for-loop-leap-years.html

- Look at all the leap years
 - Turn boxes blue and red



while loop

- Next most common loop

start

```
while( stop test ){  
    Everything between the {} is in the loop.  
    BEST PRACTICE: ALWAYS USE {}  
    move on  
}
```




while loop

```
var start = 0;
var stop = 10
var idx = start;

while(idx < stop){
  console.log(idx);
  idx++;
}
```



while loop challenge 01 10 minutes

- 03-loops
 - [while-loop-challenge01-begin.html](#)

```
while("stop test"){  
  console.log(idx);  
  "move on"  
}
```

- Replace “stop test” and “move on” with valid JavaScript
 - See previous slide
 - Use provided variables

jQuery

Interacting with HTML Input Elements





Interacting with HTML

- DOM
 - Document Object Model
 - How JavaScript “sees” the page



alterdom01.html

- Hello Code 2 - Exercises
 - Logic Ternary

Final point

A one-line description of it



Weather

Interacting with HTML Input Elements



“This is a super-important quote”



- From an expert

**This is the most
important takeaway
that everyone has to
remember.**

—



Thanks!

Contact us:

Your Company
123 Your Street
Your City, ST 12345

no_reply@example.com
www.example.com

