

# JavaScript Juggernauts

Web Development Boot Camp  
Lesson 3.3



# Today's Class

# Objectives

---

In today's class, we'll cover:



JavaScript Functions



JavaScript Objects



Building Simple JavaScript Applications

# JavaScript Functions

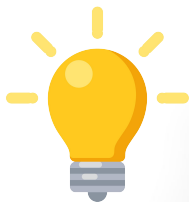


# Instructor Demonstration

## Logging: No Functions

# Mondo Repetitive

Who wants to maintain this?



**Hint:** No one.



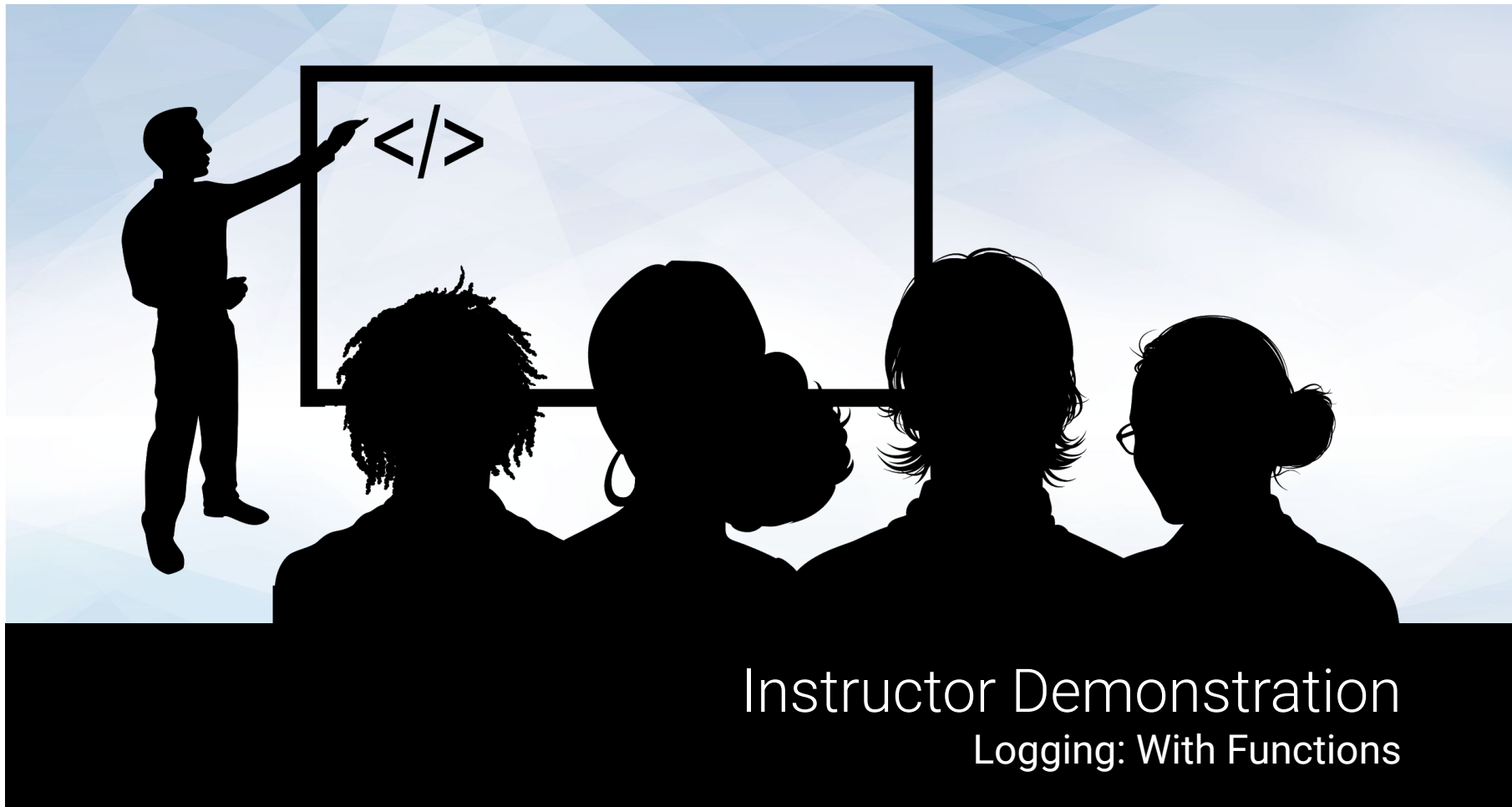
```
// For Loop for Brands
for (var i = 0; i < brands.length; i++) {
  console.log(brands[i]);
}
console.log("-----");

// For Loop for Heroes
for (var i = 0; i < heroes.length; i++) {
  console.log(heroes[i]);
}
console.log("-----");

// For Loop for booksOnMyShelf
for (var i = 0; i < booksOnMyShelf.length; i++) {
  console.log(booksOnMyShelf[i]);
}
console.log("-----");

// For Loop for thingsInFrontOfMe
for (var i = 0; i < thingsInFrontOfMe.length; i++) {
  console.log(thingsInFrontOfMe[i]);
}
console.log("-----");

// For Loop for howIFeel
for (var i = 0; i < howIFeel.length; i++) {
  console.log(howIFeel[i]);
}
console.log("-----");
```



# Instructor Demonstration

## Logging: With Functions

# Much Better with Functions!

---

Squeaky clean code. Minimal repetition.

```
// Here we create a "Function" that allows us to "call" (run) the loop for any array we wish.  
// We pass in an array as an "argument".  
function consoleInside(arr) {  
  
    // We then loop through the selected array.  
    for (var i = 0; i < arr.length; i++) {  
  
        // Each time we print the value inside the array.  
        console.log(arr[i]);  
    }  
    console.log("-----");  
}
```





# Partner Activity:

## My First Functions

**Suggested Time:**  
20 minutes



# Partner Activity: My First Functions

---



Working in pairs and using the starter file sent to you, fill in the missing functions and function calls.



**Note:** Try to finish all four functions if you can, but don't worry if you only get one or two. The important thing is that you completely finish at least one function.



**HINT:** Look back to the previous example if you need help.

**Suggested Time:** 20 minutes



# JavaScript Objects



# Instructor Demonstration

## Good Arrays



# Instructor Demonstration

Joan of Arc (Bad Arrays)

# Associated Data ==/= Arrays

---

Relating two separate arrays is not fun.

```
var joanOfArcInfoParts = ["Real Name", "Grew Up Where", "Known For", "Scars", "Symbolism"];

var joanOfArcInfoValues = ["Jehanne la Pucelle.", "Domremy, a village in northeastern France.",
    "Peasant girl, daughter of a farmer, who rose to become Commander of the French army.",
    "Took an arrow to the shoulder and a crossbow bolt to the thigh while trying to liberate Paris.",
    "Stands for French unity and nationalism."];
```



# Instructor Demonstration

## Gandalf the Grey Objects

# Gandalf: The Object

Gandalf's **properties** and **values** are associated in object form, making it easy to recall specific data.

```
11  var gandalf = {
12      "real name": "Gandalf",
13      "age (est)": 11000,
14      "race": "Maia",
15      "haveRetirementPlan": true,
16      "aliases": [
17          "Greyhame",
18          "Stormcrow",
19          "Mithrandir",
20          "Gandalf the Grey",
21          "Gandalf the White"
22      ]
23  }
24
25  // Object properties can be accessed with "bracket notation"
26  alert("My name is " + gandalf["real name"]);
27
28  // Or with "dot notation" if the property has no spaces
29  if (gandalf.haveRetirementPlan) {
30
31      // Or with a variable that matches the name of the property
32      var ageProperty = "age (est)";
33      var years = gandalf[ageProperty];
34      alert("My 401k has been gathering interest for " + years + " years!");
35  }
```



# Objects Visualized

This is Gandalf. According to code, Gandalf is an **object**.

var gandalf	=	{
-------------	---	---



"real name"	:	"Gandalf"	,
-------------	---	-----------	---

"age (est)"	:	11000	,
-------------	---	-------	---

"race"	:	"Maia"
--------	---	--------

}
---

# Objects Visualized

These are Gandalf's **properties** (like descriptors).

var gandalf	=	{
-------------	---	---



"real name"	:	"Gandalf"	,
-------------	---	-----------	---

"age (est)"	:	11000	,
-------------	---	-------	---

"race"	:	"Maia"
--------	---	--------

}
---

# Objects Visualized

These are the **values** of Gandalf's properties.

var gandalf	=	{
-------------	---	---



"real name"	:	"Gandalf"	,
-------------	---	-----------	---

"age (est)"	:	11000	,
-------------	---	-------	---

"race"	:	"Maia"
--------	---	--------

}
---

# Objects Visualized

Thus: `gandalf["race"] = "Maia"`

`var gandalf`

`=`

`{`



`"real name"`

`:`

`"Gandalf"`

`,`

`"age (est)"`

`:`

`11000`

`,`

`"race"`

`:`

`"Maia"`

`}`



# Instructor Demonstration

## Gandalf: The Grey Objects (Repeat)



## **Group Activity** (2 people): Basic Objects

**Suggested Time:**  
15 minutes



# Group Activity: Basic Objects

---



With a partner, spend a few minutes studying the code just slacked to you.



Then below each comment, write code to log the relevant information about the provided `car` object.



**Bonus:** If you finish early, create a new object of your own. Slack out a snippet of the code to the class when you are done. Be creative!

Suggested Time: 15 minutes



A close-up, high-angle shot of a computer keyboard. The central focus is a large, white, rectangular key with rounded corners. On this key, there is a dark blue icon of a coffee cup with three wavy lines above it representing steam. Below the icon, the word "Break" is printed in a dark blue, serif font. The key is set against a light-colored, textured keyboard surface. Surrounding the main key are other keys, including one with a double quote symbol to the left and one with a dash/slash symbol to the right, all in a similar white and blue color scheme.

Break





# Instructor Demonstration

## Run That Car!



## **Challenge:** Run That Car!

**Suggested Time:**  
15 minutes



# Challenge: Run That Car!

Using the code from the previous activity as a starting point, create a complete application that fulfills the following requirements:



Users can enter keyboard input (letters).



Each of the car's methods are assigned to a key.



When the user presses a key, it calls the appropriate function.



These letters also trigger a global function called `rewriteStats()` that logs the car's make, model, color, mileage, and `isWorking` status to the console.

Suggested Time: 15 minutes





# Questions?