

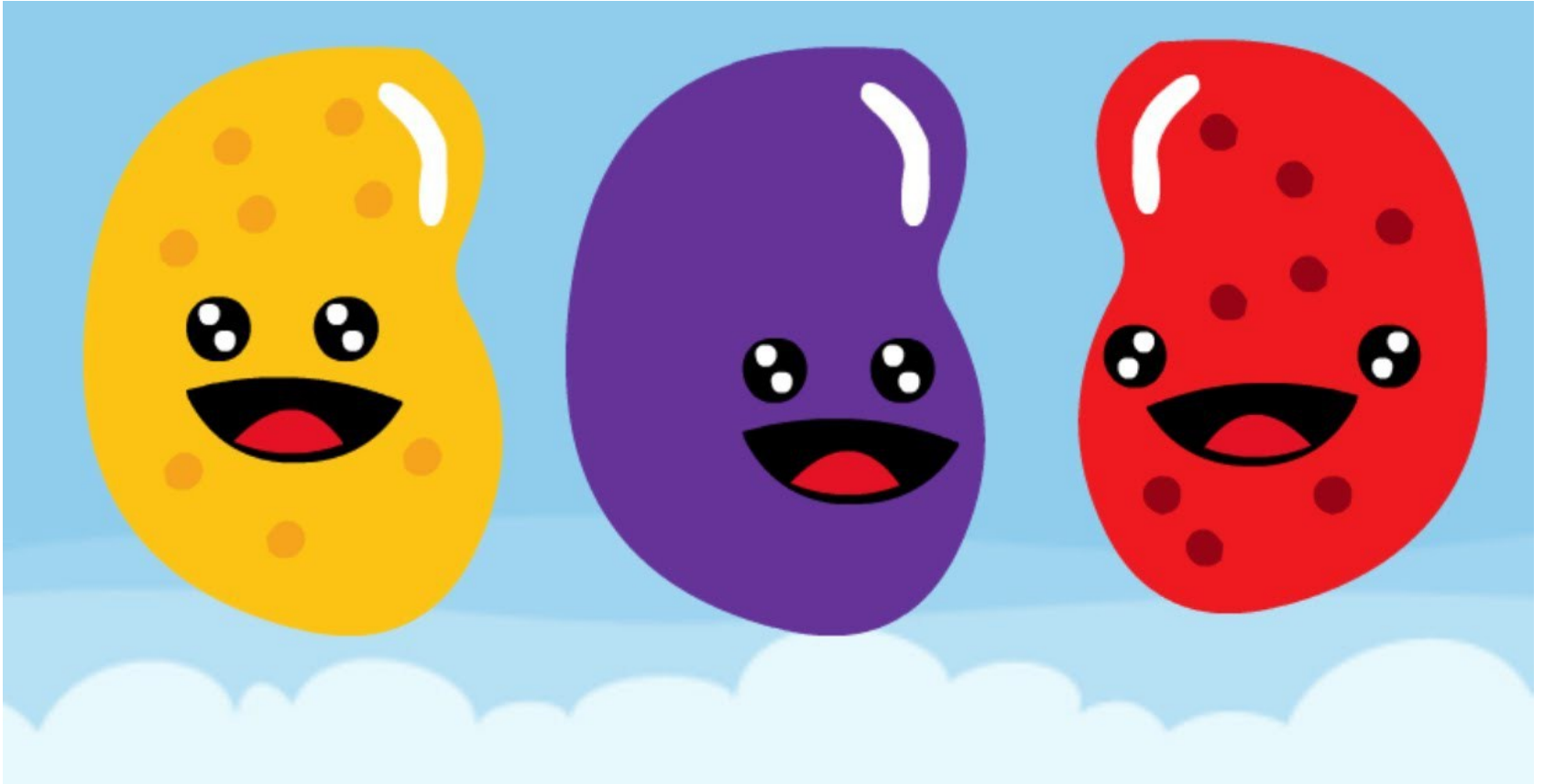
JavaScript Juggernauts

This will soon be
you...



***JavaScript
Juggernauts.***

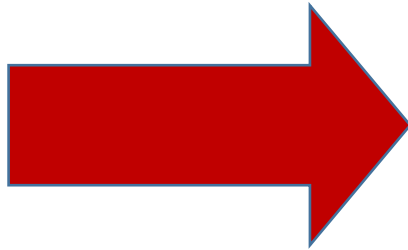
But right now...



Maybe feeling like

JavaScript Jellybeans.

Transformation to Come



HANG IN THERE!

Objects

Demo Time

Instructor: Demo
(*GoodArray.html* | 28-GoodArray)

Demo Time

Instructor: Demo

(JoanOfArcArrays.html | 29-JoanOfArcArrays)

Associated Data

==/= Arrays

```
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."];
```

**Relating two separate
arrays is not fun.**

Demo Time

Instructor: Demo

(gandalf-the-grey-objects.html | 30-GandalfTheGreyObjects)

Gandalf – The Object

```
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  // Object properties can be accessed with "bracket notation"
25  alert("My name is " + gandalf["real name"]);
26  // Or with "dot notation"
27  if (gandalf.haveRetirementPlan) {
28      // Or with a variable that matches the name of the property
29      var ageProperty = "age (est)";
30      var years = gandalf[ageProperty];
31      alert("My 401k has been gathering interest for " + years + " years!");
32  }
33  // You can access arrays and their properties from an object
34  alert("I have more than " + gandalf.aliases.length + " aliases");
35  // Non-existent properties return undefined
36  alert("My designation is " + gandalf["designation"]);
```

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

Objects Visualized

```
var  
gandalf = {
```



“real name”

:

“Gandalf”

,

“age (est)”

:

11000

,

“race”

:

“Maia”

}

This is Gandalf. According to code... Gandalf is an **Object**.

Objects Visualized

```
var  
gandalf = {
```



“real name”

:

“Gandalf”

,

“age (est)”

:

11000

,

“race”

:

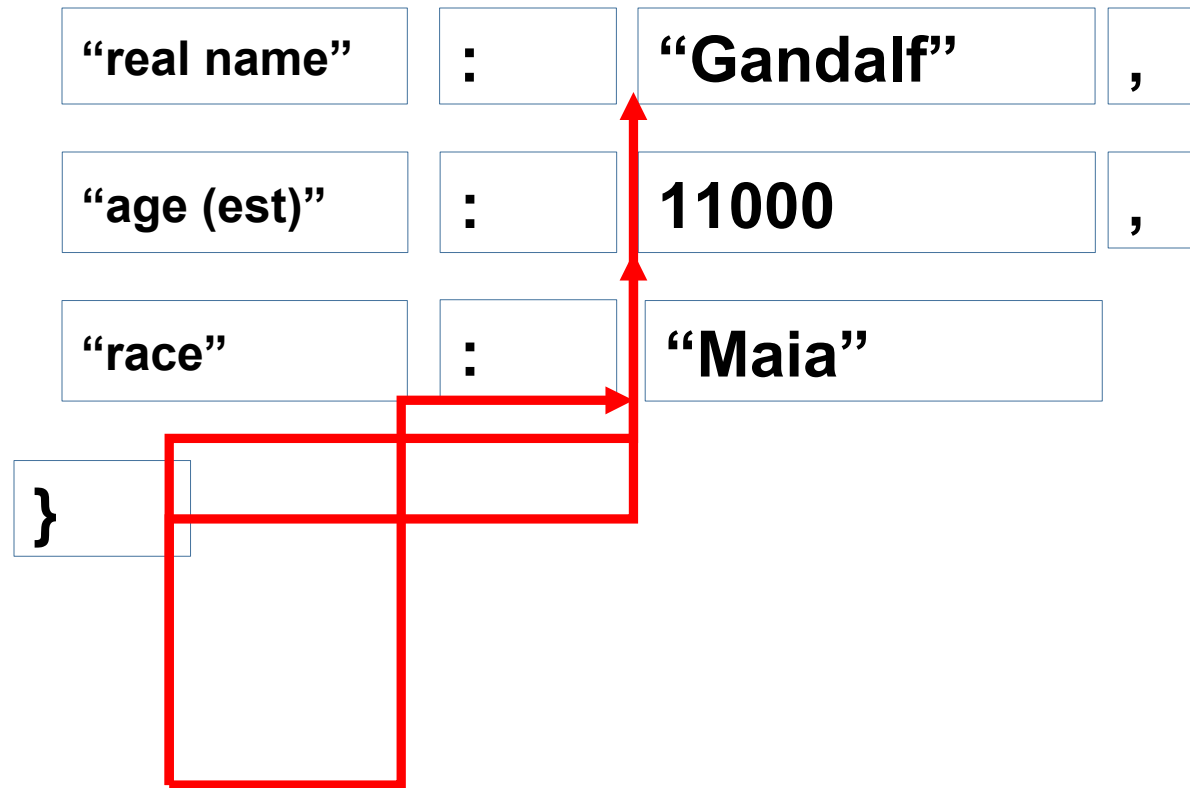
“Maia”

}

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

Objects Visualized

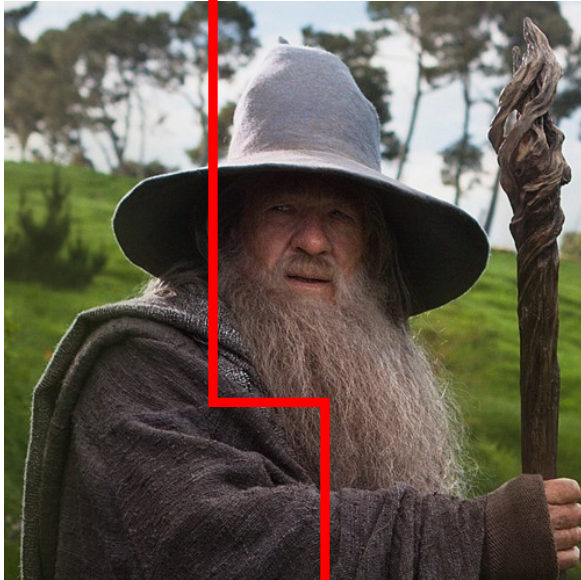
```
var  
gandalf = {
```



These are the “**values**” of Gandalf’s **properties**.

Objects Visualized

```
var gandalf = {
```



“real name”

:

“Gandalf”

,

“age (est)”

:

11000

,

“race”

:

“Maia”

}

Thus: `gandalf[“race”] = “Maia”`

Demo Time

Instructor: Repeat Demo

(gandalf-the-grey-objects.html | 30-GandalfTheGreyObjects)

Code Dissection / Creation: Basic Objects

- With a partner, spend the next few moments studying the code just slacked to you.
- Then, write code below each comment to log the relevant information about the provided car object.
- **Bonus:** If you finish early, create a brand new object of your own. Slack out a snippet of the code to the class when you are done. Be Creative!

Demo Time

Instructor: Demo in Browser

(carGame_Solved.html | 32-CarGame)

Code Creation: Run that Car!

- Using the code from the previous activity as a starting point, create a complete application such that:
 - 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.
 - **HINT:** You will need to use the `document.onkeyup()` function to collect input from the user's keyboard.

Questions
