JS INTRO

Values, Variables, and All That

SWBAT

- **Utilize variables** to **name and reuse** values
- **Discuss and use** collections to group and structure values: **Arrays**
- **USE SOME CONTROL FLOW**

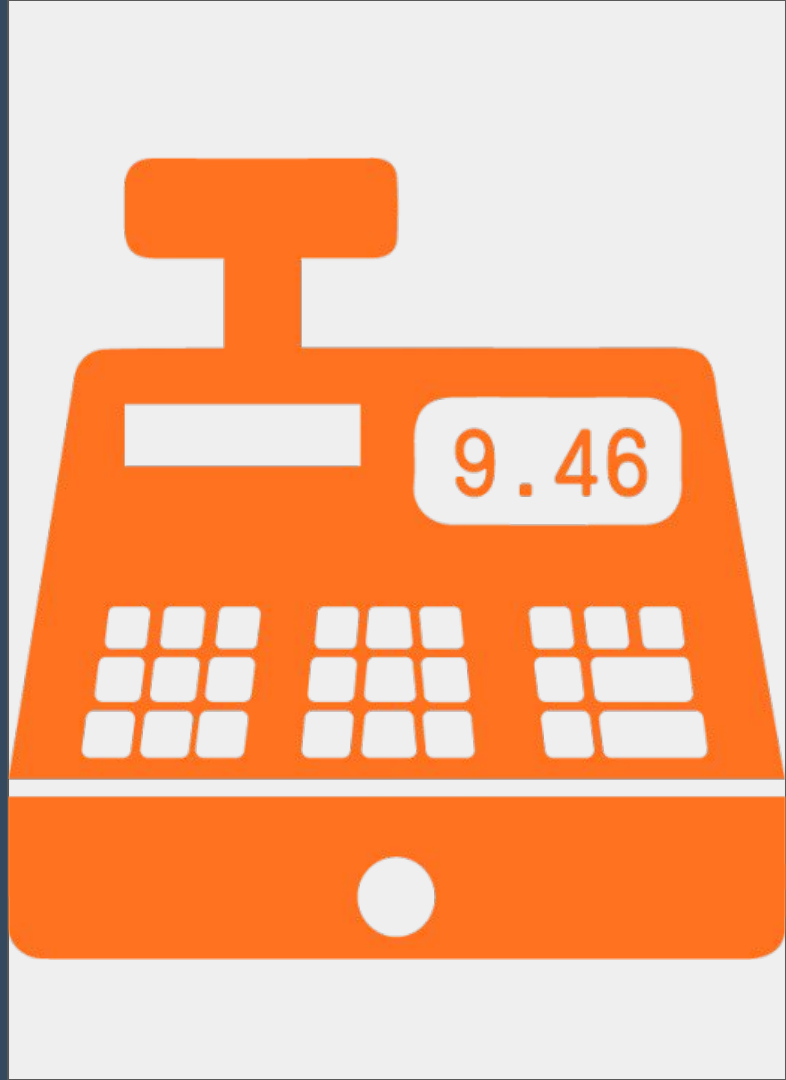
Learning Languages

- Built-in (primitive) values/types
- Means of Combining values and Control flow structures
- Means of Abstracting

Values

We begin by getting to know the various types of values:

- Numbers
- Strings
- Booleans



Values

Go to REPL.it

- Select JavaScript
- And Try the following in the native JavaScript Browser

Repl.it is a great resource for fiddling with JS topics.

```
1
// => 1
2.2
// => 2.2

"hello"
// => "hello"

true
// => true
false
// => false
```

Expressions

Once we feel comfy we using these values it’s time to learn how to build up simple expressions.

```
1 + 1
// => 2
2.2 + 3
// => 5.2

"hello" + "world"
// => "helloworld"

true || false
// => true
false && true
// => false
```

Exercise

For each value from 1 to 10

- Calculate the number times itself

Which values end in a number that was the original value.

What happens if you square those values?

```
1 * 1
// => ?
2 * 2
// => ?

...
```

Exercise

Add up the following values

1/3 + 1/9 + 1/27 + 1/81

What do you think this would add up to if we did this forever?

```
1/3 + 1/9 + 1/27 + 1/81

// => ?
```

Exercise

The interwebs love to debate this expression

```
6/2(1+2)

// => ?
```

Exercise

The point here is to just use parens.

```
((6/2)*(1+2))

// => 9
```

Our First Abstraction

NAMING


User_Profile
user_profile
userProfile
usersProfiles
userprofile
UserProfile

NAMING

What is this?

NAMING

Some might want to call it a chair

NAMING

Others might want to call it a drum

NAMING

Few realize that is actually just a shoe or stilt

NAMING

The right name requires insight into both what the thing is and how it will be used.

```
num
vs
age
vs
studentAge
```

Variables

- If you have a value then give it a name!
  - Makes its role clear
  - Re-use its value
- Use the `var` keyword to DECLARE a new variable name

```
// Reduces technical debt
var studentAge = 21;


studentAge
// => 21
```

Variables

Nice and
DESCRIPTIVE names

```
var userName;
var favBook;
var count;
```

Variables

Non-descriptive

variable names

```
var a;
var b;

var i;
var j;
var k;
```

NOT BAD
FOR looping

Not bad for looping, but good luck reading them later

Variables

Non-descriptive

variable names

```
var a;
var b;


var i;
var j;
var k;
```

a, b, c and x, y, z can be okay quick function params

NOT BAD FOR looping

Not bad for looping, but good luck reading them later

Variables

```
// Declare and initialize
var userName = “john”;

// Declare
var favColor;
// Then initialize
favColor = blue;
```

Variables

```
// Declare and initialize
var userName = “john”;

// Declare
var favColor;
// Then initialize
favColor = “blue”;
```

```
// Declare and initialize
//  multiple variables
var userName = “john”,
    favColor = “blue”;
```

Variables

- In your console create a variable for your first name and another variable for your last name.
- In your console create a variable for you full name that holds the sum of your first and last name with space separating them. Use the variables you created for the first and last name.
- Pair: using only 1 new variable swap the values of your first and last name

Arrays

An array is a collection of values indexed by numbers.

```
var friends = ["jane", "john"];

friends[0]
// => "jane"

friends[1]
// => "john"
```

Arrays

You can update its indexed values

```
var friends = ["jane", "john"];

friends[0] = "jess";

friends
// => ["jess", "john"]
```

Arrays

You can add and remove values using `push`, `pop`, `shift`, and `unshift`

```
var favCars = ["jag", "benz"];

favCars.push("ford");
// => ["jag", "benz", "ford"];

favCars.pop();
// => "ford"
```

Arrays

- Create a variable for an array with the two most popular **website names.**
  - Add the third and fourth most popular sites using the methods discussed
  - Remove the **most** popular site from the list
  - Add your favorite site to the **front** of the list
  - **Google it:** copy out all but the first and last values
  - **Google it:** remove the **second** value in the **list**

LOOPS

- A `for` loop is the most kind of loop you will use

```
var favCars = ["jag", "benz", "ford", "tesla"];


for (var index = 0; index < favCar.length; index += 1) {
  console.log(favCars[index])
}
```

LOOPS

- A `for` loop is the most kind of loop you will use

```
var favCars = ["jag", "benz", "ford", "tesla"];


for (var index = 0; index < favCar.length; index += 1) {
  console.log(favCars[index])
}
```

The initializing of the loop

LOOPS

- A `for` loop is the most kind of loop you will use

```
var favCars = ["jag", "benz", "ford", "tesla"];


for (var index = 0; index < favCar.length; index += 1) {
  console.log(favCars[index])
}
```

The condition to terminate the loop

Loops

- A `for` loop is the most kind of loop you will use

```
var favCars = ["jag", "benz", "ford", "tesla"];


for (var index = 0; index < favCar.length; index += 1) {
  console.log(favCars[index])
}
```

The block to run each iteration

LOOPS

- A `for` loop is the most kind of loop you will use

```
var favCars = ["jag", "benz", "ford", "tesla"];


for (var index = 0; index < favCar.length; index += 1) {
  console.log(favCars[index])
}
```

The statement to run to proceed to the next iteration

LOOPS

- A `for` loop is the most kind of loop you will use

```
var favCars = ["jag", "benz", "ford", "tesla"];


for (var index = 0; index < favCar.length; index += 1) {
  console.log(favCars[index])
}
```

The statement to check to terminate iteration

LOOPS

- A `for` loop is the most kind of loop you will use

```
var favCars = ["jag", "benz", "ford", "tesla"];


for (var index = 0; index < favCar.length; index += 1) {
  console.log(favCars[index])
}
```

The block to run this iteration

LOOPS

- A `for` loop is the most kind of loop you will use

```
var favCars = ["jag", "benz", "ford", "tesla"];


for (var index = 0; index < favCar.length; index += 1) {
  console.log(favCars[index])
}
```

The statement to run to proceed to the next iteration

Loops

- Create a variable for an array with the 5 most popular **website names.**
  - Print the values in reverse order
- Pair: iterate through the 5 values and use swapping to reverse the values.

Conditionals

- An `if` allows you to specify a condition to run a block of code

```
var points = 100;

if (points > 99) {
  console.log("you win!");
}
```

Conditionals

- An `if` allows you to specify a condition to run a block of code
  - You can specify a default block of code to run otherwise using `else`

```
var points = 98;

if (points > 99) {
  console.log("you win!");
} else {
  console.log("KEEP PLAYING!");
}
```

Conditionals

- An `if` allows you to specify a condition to run a block of code
  - You can specify a default block of code to run otherwise using `else`
  - An `else` followed by `if` allows you specify another condition

```
var points = 98;

if (points > 99) {
  console.log("you win!");
} else if (point === 98) {
  console.log("ALMOST THERE!");
} else {
  console.log("KEEP PLAYING!");
}
```

Conditions

- Create a variable with an array of 10 different ages
  - Loop through
    - print “You can drink” if they are over 21
    - print “you can barely drink” if they are 21
    - print “you ARE NOT allowed to drink” if they are less than 21

REVIEW

- We used variables to name and swap values
- We used `for` loops to iterate through collections
- We used Conditions to check determine what kind of message to print

NEXT...

- We want to utilize arrays and variables to create a fun application using events and selectors