# JS CONTROL FLOW

Primitives, Variables, and All That

# SWBAT

- **Utilize variables** to **name and reuse** values
- **Discuss and use** collections to group and structure values: **Objects and Arrays**
- **Review and utilize** event listening and DOM selection

# Variables

- If you have a value then give it a name!
  - Makes its role clear
  - Re-use its value

# 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

# 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

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

Nice and **DESCRIPTIVE** names

# 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

```
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

- 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

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

```
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

- 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

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

Nice and **DESCRIPTIVE** names

```
var a;
var b;
var i;
var j;
var k;
```

Not all bad, 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”
```

```
var friends = [“jane”, “john”];

friends[0] = “janice”;
// => “jane”

friends
// => [“janice”, “john”];
```

# Arrays

- An array is a collection of values indexed by numbers.
  - 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

# Assessment

https://jsbin.com/batixaw/edit?html,css,js,output

# Assessment

- Create an index html with a container div ID’ed as “pixel-canvas”.
- Link an external JS file
  - Be sure to verify it’s linked properly
- Add an external CSS file

# Assessment

- In your CSS give the #pixel-canvas div a set width and height
  - Try to make it square
- Add CSS to to make each pixel have equal height and width of `10px`
  - Give each a `float` of `left`
  - Give each a border
- Create a class called “filled-black” with background-color black

# Assessment

- Create an array of div elements with a class pixel and with 400 elements called `pixels`
  - `["<div class='pixel'></div>", "<div class='pixel'></div>", … ]`
- Hint:

```
var pixel = "<div class='pixel'></div>" ;
var pixels = [];

for (var count = 0;  count < 400; count += 1 ) {
    pixels.push(pixel);
}
```

# Assessment

- In the console try out the `join` method on the array
- Select the `"pixel-canvas"` element by id and set its innerHTML to the pixels joined together

# Assessment

- Grab the pixel canvas element and try the following

```
var pixelCanvas = document.getElementById( "mousemove");

pixelCanvas.addEventListener( "mousemove", function (event) {
  var pixel = event.target;

  pixel.classList.add( "filled-black");
});
```

# Assessment

- Listen for a “mousedown” and “mouseup” event on the “pixel-canvas” element

```
var pixelCanvas = document.getElementById( "pixel-canvas");
var pressed = false;

pixelCanvas.addEventListener( "mousedown", function (event) {
  console.log(event.target);
  pressed = true;
});

document.addEventListener( "mouseup", function (event) {
  console.log(event.target);
  pressed = false;
});
```

# Assessment

- Listen for a “mousemove” event on the “pixel-canvas” element
  - If `pressed` is `true` add the filled-black to classList for the `event.target`

```
pixelCanvas.addEventListener("mousemove", function (event) {
  var pixel = event.target;

  if (pressed) {
    pixel.classList.add("filled-black");
  }
});
```

# Assessment

- Create a button called `erase` with id of `drawing-mode`
  - When clicked change the innerHTML to say draw
  - Set a variable called `drawingMode` to “erasing”
- If drawingMode is set to “erasing” remove the classList “pixel-filled”
- If the erase button is clicked when drawingMode is set to erasing
  - Change the innerHTML to say “erase”
  - Change the `drawingMode` to “drawing”