IA FOR THE WEB

JavaScript Part 2
Fall 2014

REVIEW

VARIABLES

- Containers for information, such as numbers, text, etc.
- Declared using **`var`** keyword

```
var counter;
counter = 1;

var counter = 1;
```

DATA TYPES

- string
- number
- boolean
- undefined
- null

IF-ELSE IF-ELSE

```
if(/* expression */){
   /*Do something. */
}else if(/* expression 2 */){
   /*Do something else. */
}else{
   /*Do this. */
}

if(page > 5){
  alert(“greater”);
}else if(page == 4){
  alert(“equal”);
}else{
  alert(“lesser”);
}
```

IF EXAMPLE

- Using an if statement to test number parity (even/odd)

IF TIPS

- Can only have one if clause
- Can only have one else clause
- if clause must go first
- else clause must go last
- Can have as many else if clauses as you need
- Only one clause will execute

WHILE

Expression
that must be true
to continue

```
while(i >= 0){
  alert(i);
  i--;
}
```

FOR

Counter

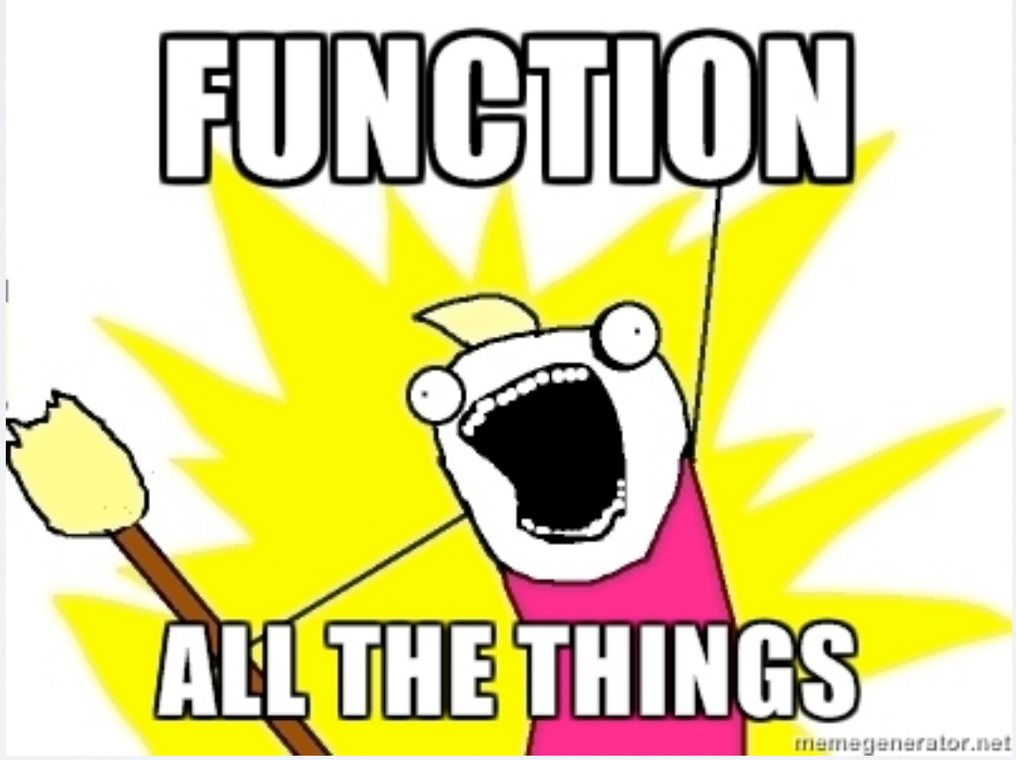
Expression
that must be true
to continue

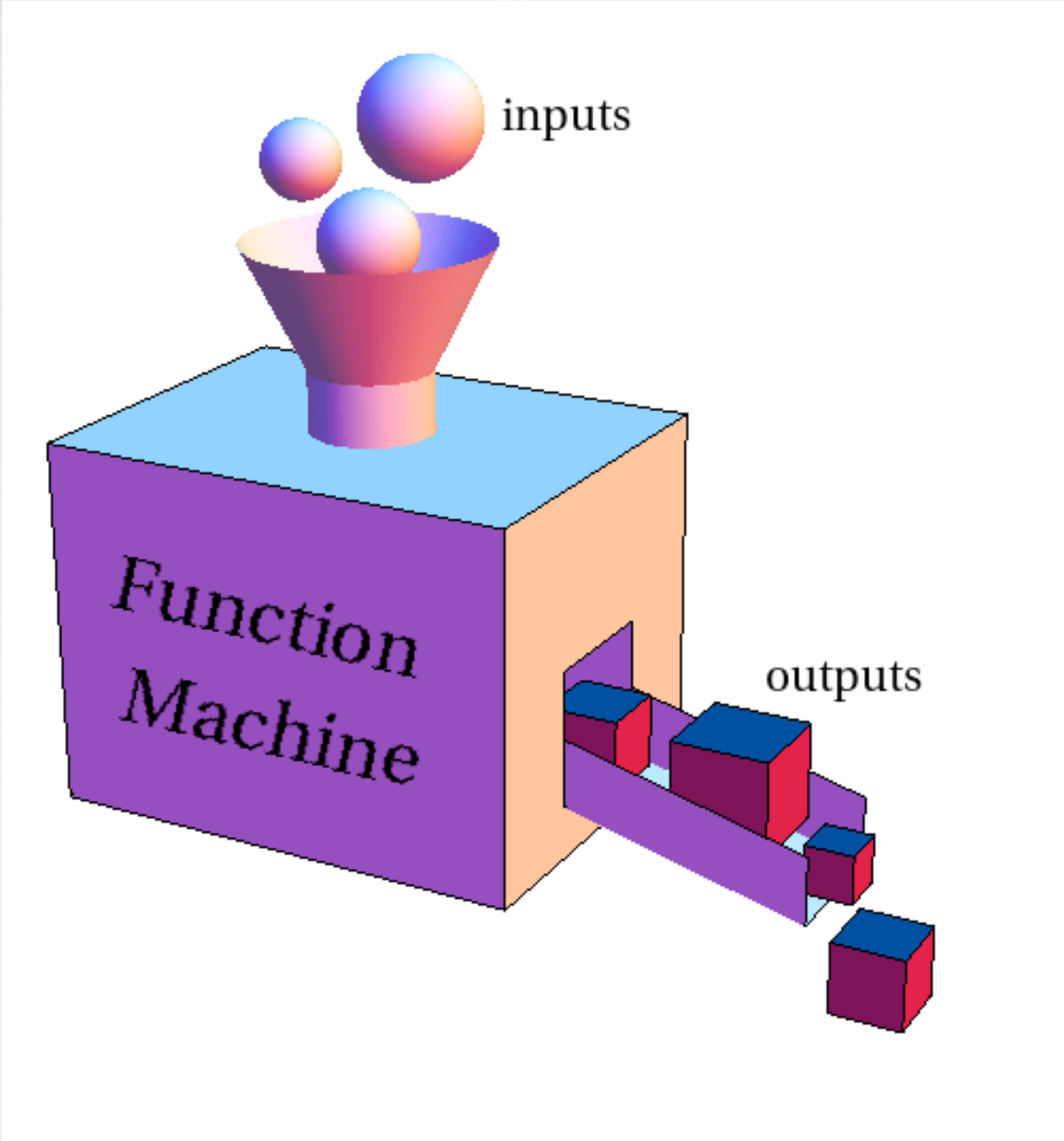
Expression that
increments or
decrements the
counter

```
for(var i = 10; i >= 0; i--){
  alert(i);
}
```

NEW STUFF

FUNCTIONS




inputs
Function
Machine
outputs

CALLING FUNCTIONS

- You need to know **at least** one thing to call a function: its name
- Example

CALLING FUNCTIONS - PARAMETERS

- Parameters let you specify input into a function
- Put parameters in the parentheses, separated by commas
- Example

CALLING FUNCTIONS - RETURNED VALUES

- Some functions return a value
- It's your responsibility to make sure that value is put into a variable if you need it
- Example

CHAINS OF FUNCTIONS

- Functions can be chained together, i.e. the result of one can be fed into another
- Example

DEFINING FUNCTIONS

Keyword indicating this is a function

Name of function

Parameters

```
function multiplyByTwo(startingNumber){

  return startingNumber * 2;

}
```

Return statement

PARAMETERS

- Can take multiple parameters. Separate them with commas:

```
function compareTwoNumbers(number1,
number2)
```

- Can take zero parameters:

```
function sayHello(){
  alert("Hello!");
}
```

RETURN STATEMENT

- Tells the function to return a value
- The value can be any type (integer, string, array, object, etc.)
- Using **`return`** by itself will return back to where the function was called without sending a value back

ARRAYS

- An array is a container for multiple pieces of data

Without Arrays	With Arrays

`var sandwich0 = “ham”;`
`var sandwich1 = “veggie”;`
`var sandwich2 = “egg salad”;`
	`var sandwiches = [“ham”, “veggie”, “egg salad”];`

ARRAY SYNTAX

- Use brackets to indicate an array
- Separate items with a comma

```
var instructors = ["Smith", "Jones",
"Thomas"];
```

ARRAYS

- Can contain different types of data in the same array
- Can contain other arrays
- Have a property, called length, that tells you how many elements it has:

```
var instructors = ["Smith", "Jones",
“Thomas"];

alert(instructors.length); //alerts "3"
```

ACCESSING ARRAY ELEMENTS

- Array elements start at 0
- Access them like this:

```
var instructors = ["Smith", "Jones",
“Thomas"];

alert(instructors[1]); //alerts "Jones"
```

ADDING TO AN ARRAY

```
var instructors = ["Smith", "Jones",
"Thomas"];

instructors[instructors.length]= "Wang";
```

LOOPING OVER AN ARRAY

```
var instructors = ["Smith", "Jones",
"Thomas"];

for(i=0; i<instructors.length; i++){

  alert(instructors[i]);

}
```

OBJECTS

- An object represents a thing
- It can contain properties and methods
- Properties are pieces of information about an object
- Methods are functions that perform an action

BOOK OBJECT

- Properties:
  - title
  - author
  - ISBN
  - UPC
  - price
- Methods:
  - open()
  - close()
  - goToPage()

PROPERTIES

- Access them using a period between the object name and the property name:

```
book.title = "Gone Girl";

alert(book.title);
```

METHODS

- Methods are called using a period as well:

```
book.open();

book.goToPage(6);
```

THE BROWSER ENVIRONMENT

- `<script>` tag
- `document`
- Document Object Model (DOM)

<SCRIPT>

- Tells the browser to load a script
- Doesn't need `@type` in HTML5
- Don't close it in the start tag:

```
<!-- Don't do this -->

<script src="filename.js"/>
```

<SCRIPT> IN <HEAD> OR <BODY>

- **`<script>`** can go in **`<head>`** or **`<body>`**
- However, JS that operates on HTML needs to have the HTML loaded first

```
<!doctype html>

<html>

<head>

<meta charset="UTF-8">

<title>JS Example 2A</title>

<script>

var heading = document.getElementById('pageTitle');

heading.textContent = "XYZ Company";

</script>

</head>

<body>

<!-- Will stay ABC Company because HTML is after script -->

<h1 id="pageTitle">ABC Company</h1>

</body>
```

INLINE SCRIPT AT THE BOTTOM OF <BODY>

```
<!doctype html>

<html>

<head>

<meta charset="UTF-8">

<title>JS Example 2B</title>

</head>

<body>

    <!-- Will change to XYZ Company because it’s after the HTML it’s changing-->

    <h1 id="pageTitle">ABC Company</h1>

    <script>

        var heading = document.getElementById('pageTitle');

        heading.textContent = "XYZ Company";

    </script>

</body>

</html>
```

@DEFER

- HTML loads first, then script runs
- Deferred scripts run in order they appear
- Not guaranteed to work on inline scripts

EXTERNAL SCRIPT AND @DEFER

```
<!doctype html>

<html>

<head>

<meta charset="UTF-8">

<title>JS Example 2C</title>

<script defer src="js-example2C.js"></script>

</head>

<body>

    <!-- Will change to XYZ Company -->

    <h1 id="pageTitle">ABC Company</h1>

</body>

</html>
```

OK, BUT WHERE SHOULD I PUT <SCRIPT>?

- You can always put it at the bottom of **`<body>`**
- If using jQuery, you can put it in the **`<head>`** and use the **`ready()`** function
- If you *know* the browser supports it, you can use **`@defer`**

DOCUMENT

- The most important object you'll work with in the browser
- Represents the HTML document
- Has a variety of methods that let you interact with the HTML

DOCUMENT OBJECT MODEL (DOM)

- Specifies what the properties and methods for **`document`** are
- Specifies properties and methods for various HTML elements
- W3Schools has a good DOM reference

GET AN ELEMENT BY ITS ID

- Use document.getElementById()
- `var city = document.getElementById('city');`
- Example

RESULT

- **`getElementById()`** returns an object whose properties and methods are specified by the DOM
- **`input`** element with **`@type=text`** is its own HTML DOM object.
- You have to know what the properties and methods are for the object you're working with

GET ALL ELEMENTS WITH A PARTICULAR TAG

- Use getElementsByTagName()
- Takes a single argument: the name of the tag
- Returns an array of all elements created with that tag

EVENTS

- So far, we've been telling the browser what to do in a step-by-step fashion
- Now we'll learn to tell it how to react when certain things happen, like key presses or button clicks
- Events are specifed by the DOM.
- More

EXERCISES

- Try the exercises at http://www.codecademy.com/tracks/javascript