

Introduction to JavaScript

Welcome

- JavaScript has many uses, but we will focus on Web Design, specifically how to add interactivity
- In this class there is an assumption that you are new to programming, but you know HTML and CSS

What you can do with It

- JavaScript is a “real” programming language
 - Store variables
 - Set decision points
 - Loop
 - Reuse code with functions
- In addition
 - Get data from the browser
 - Manipulate the DOM that browsers use to create web pages

Variables

- Store data and refer back to it later

Decision Points

- Use control statements to decide which code to run under different circumstances

Looping

- Avoid writing the same (or similar) code over and over again
- Determine at runtime how many times you want to run some code

Functions

- Reuse code multiple times, but only write it once
- Use code from others

Manipulating the DOM

- JavaScript can find, add, and delete elements from the DOM
- Can also react to mouse clicks, page reloads, and other actions

Review

- A major component of learning any programming language is practice and repetition
- Expect to make mistakes
 - if you aren't you aren't learning

© Colleen van Lent
University of Michigan
School of Information

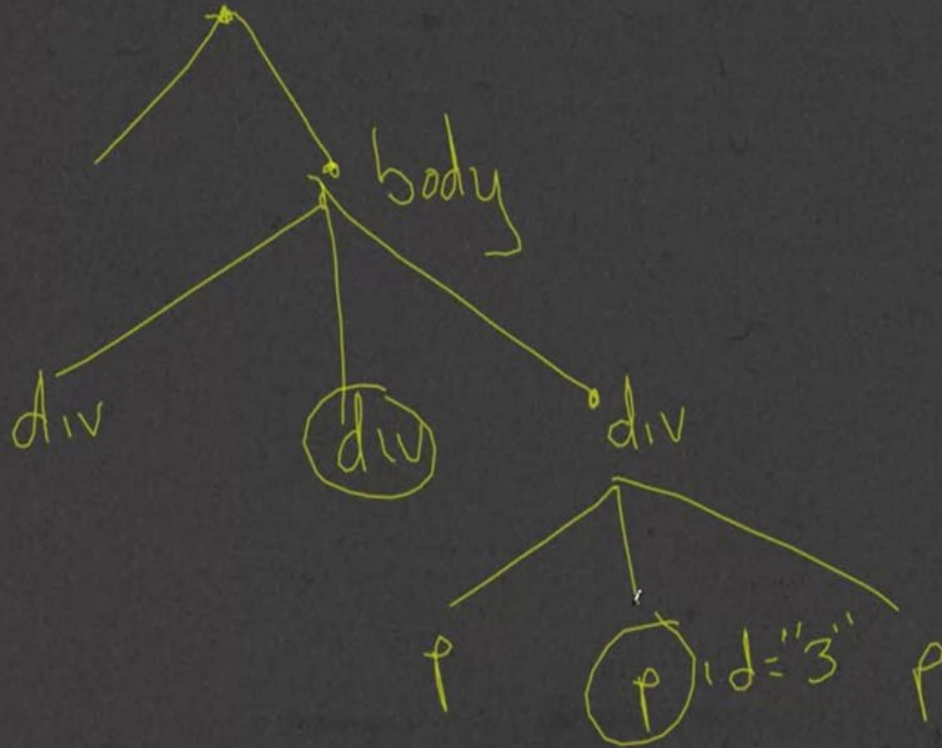
Unless otherwise noted, this work is licensed under the
CC BY-NC 4.0 license.

DOM Review with Object Oriented Programming

Web Pages are built upon the DOM

- Document Object Model
- Structures documents like a tree
- Every node has one parent, and possibly many children
- Nodes have properties, methods, and events

Sample Document



The DOM and JavaScript

- Page content is represented by the DOM
- Scripting languages (JavaScript) use the DOM to interact with the document

How Does It Work?

- Accessing the DOM is done with an API – Application Programming Interface
 - No matter which browser, no matter which scripting language, the API is the same

The DOM objects/elements

- **document** – the root of the page
 - `document.URI`, `document.height`, `document.links`, `document.bgColor`,....
- **element** – a node in the tree
 - Returned by a member of the API
- **nodeList** – an array (group) of elements
 - `document.getElementsByTagName('p')` would return a set of nodes
- **attribute**
 - A node in the DOM, though rarely used that way. Another way to manipulate/change the document

Specific APIs

- `document.getElementById(id)`
- `document.getElementsByClassName(class)`
- `element.innerHTML`
- `element.style`
- `element.setAttribute(attribute, value)`
- `element.removeAttribute(attribute)`

Review

- As you learn more JavaScript, you will be able to use the APIs
- We will start slow, but the important part is to eventually feel comfortable searching for these tools

© Colleen van Lent
University of Michigan
School of Information

Unless otherwise noted, this work is licensed under the
CC BY-NC 4.0 license.

Newer DOM methods

querySelector and querySelectorAll

Selecting the First Element

- `getElementById()` takes a single parameter and that parameter must be an id selector
- `querySelector()` Method
 - returns first result of the given selector which could be *anything*— except pseudo-elements
- Because the selector can be anything which is a valid CSS selector you must include the *#*, *.*, etc.

Selecting Multiple Elements

- The `querySelectorAll()` method is identical to the `querySelector()` but returns all the found values
- Again, while `getElementsByClassName` doesn't need the "." as part of the selector, `querySelectorAll` does

Deciding on a Method

- Speed won't be an issue for you
- `querySelector` allows you to use any css selector
- I am less prone to typos in the method name
- `getElementById`, `getElementsByClassName`, etc have more mnemonic names.
- I am less prone to typos in the css selector

© Colleen van Lent

University of Michigan

School of Information

Unless otherwise noted, this work is licensed under the
CC BY-NC 4.0 license.

Output

Interactivity

- HTML5 and CSS3 are not really interactive
- New elements and pseudo-classes can only go so far

What Can JavaScript do?

- Read and write HTML elements
- Reacts to events (mouse events, keyboard events, etc.)
- Validate data
- Detect the visitor's browser
- Create cookies

JavaScript Output

- JavaScript doesn't have a built-in print function
- Data is displayed via
 - an alert box using *window.alert()*
 - *a prompt using window.prompt()*
 - HTML output using *document.write()*
 - HTML element using *innerHTML()*
 - the browser console using *console.log()*

alert()

- In JS, an alert is a pop-up window that displays information
- The parentheses mean that this is a function

```
alert("My Message Here")
```


prompt()

- Very similar to alert, but wants input.

```
prompt("Enter your name: ")
```

document.write()

- What if we want something permanent?
- document.write() writes directly to the page
- Here we have combined a function with an object that will add to page

```
document.write("Time to learn JavaScript")
```

document.write()

- Not usually recommended since it can easily be misused

innerHTML

- To change the contents of the DOM, use innerHTML combined with the element you want to change

no parentheses!!



```
element.innerHTML = "Time to learn JavaScript"
```

console.log()

- This option write the data to the browser console
- The console is a place to see what is going on during the execution of your program

```
console.log("Leave a secret message")
```

The console

- You should be utilizing the console by now
- Does more than take “print” statements, also provides debugging information for JavaScript, HTML and CSS

Debugging

- **Safari:** Preferences → Advanced Check the Show development menu in menu box”
- **Google Chrome:** Developer → JavaScript Console
- **Firefox:** Tools → Console
- **Edge:** F12

Review

- Right now, we are doing simple things with output
- As you learn more, the power grows

© Colleen van Lent
University of Michigan
School of Information

Unless otherwise noted, this work is licensed under the
CC BY-NC 4.0 license.

Variables

Storing Data

- Part of learning to program is learning to store data
- In JavaScript, data is stored in variables
- To use a variable, you have to declare it



Storing Data

Variable Name	Memory Location	Value “stored” in computer
name	11001100001101	“Christopher”
age	11001100001110	
...	11001100001111	
...	11001100010000	
...	11001100010001	
...	11001100010010	
...	11001100010011	

Variable Names

- Consists of letters, digits, underscores, and dollar sign(\$)
- Can not start with a digit
- Are case-sensitive...
 - name Name, naMe, NAME are all different variables
- Should be mnemonic (meaningful)

Variable assignments

- It is silly to have a variable if you are never going to use it
- You can assign values using the = operator

```
var name ="Colleen"
```



assignment operator

Assignment statements

- I like to refer to the LHS and RHS of statements
- LHS – the variable being updated
- RHS – the new value that will be stored in the variable

```
var name
```

```
name = "Colleen"
```

↑
LHS

↑
RHS

Using a Variable

```
var name = prompt("What is your name?")  
document.write(name)  
var date= Date()  
document.write(date)  
var location= window.location  
document.write(location)
```

Review

- Variables are a key component of creating interactive programs
- We will be using them in the remaining lectures so practice them and feel comfortable

© Colleen van Lent
University of Michigan
School of Information

Unless otherwise noted, this work is licensed under the
CC BY-NC 4.0 license.



Data Types

Assignments

```
var name = prompt("What is your name?")
```

```
var name = Date()
```

```
var name = window.location
```

Types

- In many programming languages, variables need to have a single type
- In JavaScript, a variable can take on many different types
- What are these types?

Number

- Numerical values
 - with or without decimals

```
var width = window.innerWidth  
var pi = 3.14
```

String

- A String is a collection of characters (letters, numbers, punctuation,)
- To create a string you put the value in quotes "..."

```
var location = window.location  
var name = "Colleen"
```

Boolean

- In programming, a boolean value is one that is either true or false

```
var status=false
```

```
var windowStatus=window.closed
```

- Later, we will learn how to write our own boolean expressions to check if things are true or false

Object

- Sometimes the variables are more complex
 - A node in the DOM a good example

```
var topic= document.getElementById("myID")
```

- Nodes are more than a single value, they have attributes

Array

- How can a function return more than one value?

```
var links= document.getElementsByTagName('a')
```

Accessing Array Elements

- Arrays store multiple value using a variable name, and an index for each element in the array

```
var links= document.getElementsByTagName('a')  
document.write(links[0])
```

- We will cover arrays in depth later in the course

Review

- Luckily in JavaScript you have a lot of flexibility with the types of data
- For now, focus on learning the types of data returned by the most common APIs

© Colleen van Lent
University of Michigan
School of Information

Unless otherwise noted, this work is licensed under the
CC BY-NC 4.0 license.

Operators and Expressions

Statements

- We have been using statements to execute our JavaScript code
- Statements often have *expressions*
- Expressions produce values

Expression

- So if you think back to $LHS = RHS$, the LHS is a variable and the RHS is what generates the value
- What are our tools for generating values on the RHS?

Assignment Operators

Operator	Example	Value stored in x
=	x = 5	5
=	y = 12 x = y	12

Arithmetic Operators

Operator	Example	Value stored in x
+	$x = 2 + 5$	7
-	$x = 5 - 2$	3
*	$x = 2 * 5$	10
/	$x = 5/2$	2.5
%	$x = 5\%2$	1

More Operators

Operator	Example	Value stored in x
++	<pre>x = 5; x++;</pre>	6
--	<pre>x = 12; x--</pre>	11
+=	<pre>x = 2; x+=5</pre>	7

String Operators

Operator	Example	Value stored in x
+	x = "Hi" + "There"	"HiThere"
+	x = "Hi" + 5	"Hi5"
+=	x = "Hi" x += "There"	"HiThere"

Boolean Operators

- We can also use operators to compare values
- Assume `x = 12`;

Operator	Example	Returns
<code>==</code>	<code>x == 5</code>	false
<code>==</code>	<code>x == 12</code>	true
<code>!=</code>	<code>x != 5</code>	true

Boolean Operators

- Assume `x = 12`;

Operator	Example	Returns
<code>></code>	<code>x > 12</code>	false
<code>>=</code>	<code>x >= 12</code>	true
<code><</code>	<code>x < 12</code>	false
<code><=</code>	<code>x <= 12</code>	true

Boolean Operators

- Assume `x = 12`;

Operator	Example	Returns...
<code>==</code>	<code>x == "12"</code>	true
<code>===</code>	<code>x === "12"</code>	false
<code>!==</code>	<code>x !== 12</code>	false

- You need to really stop and think about these operators...

Logical Operators

- Assume `x = 12`;

Operator	Example	Returns...
<code>&&</code>	<code>(15 > x) && (x > 5)</code> both sides must be true	true
<code> </code>	<code>(15 > x) (x > 5)</code> at least one side must be true	true
<code>!</code>	<code>!(x == 12)</code>	false

Review

- Programming is not just about knowing the syntax of a language
- You need to think about the logic behind what you want to do, before you start to code

© Colleen van Lent
University of Michigan
School of Information

Unless otherwise noted, this work is licensed under the
CC BY-NC 4.0 license.