Intro to Web Programming

Clarissa Littler

August 30, 2016

What we'll be covering?

- A short review of HTML and CSS
- A bare bones introduction to JavaScript
- Examples of how to use JavaScript to alter web sites

What we'll be covering?

- A short review of HTML and CSS
- A bare bones introduction to JavaScript
- Examples of how to use JavaScript to alter web sites

What we'll be covering?

- A short review of HTML and CSS
- A bare bones introduction to JavaScript
- Examples of how to use JavaScript to alter web sites

What we'll be covering?

- A short review of HTML and CSS
- A bare bones introduction to JavaScript
- Examples of how to use JavaScript to alter web sites

What we won't cover

- Most of JavaScript
- How to write a server
- How HTML and CSS work
- How to use frameworks to build a site

What we won't cover

- Most of JavaScript
- How to write a server
- How HTML and CSS work
- How to use frameworks to build a site

What we won't cover

- Most of JavaScript
- How to write a server
- How HTML and CSS work
- How to use frameworks to build a site

What we won't cover

- Most of JavaScript
- How to write a server
- How HTML and CSS work
- How to use frameworks to build a site

The point of this course

- Start you on the right track
- Give a taste for what web-programming is
- Explain the basic pieces

The point of this course

- Start you on the right track
- Give a taste for what web-programming is
- Explain the basic pieces

The point of this course

- Start you on the right track
- Give a taste for what web-programming is
- Explain the basic pieces

The point of this course

- Start you on the right track
- Give a taste for what web-programming is
- Explain the basic pieces

Client and server

Two pieces that talk to each other to make a site

Server

- Sends data to the browser
- Saves information for long term use
- Receives requests from the client

Client

- Receives data from the server
- Renders server data into a usable page
- Handles the user interface

Client and server

Two pieces that talk to each other to make a site

Server

- Sends data to the browser
- Saves information for long term use
- Receives requests from the client

Client

- Receives data from the server
- Renders server data into a usable page
- Handles the user interface

Client and server

Two pieces that talk to each other to make a site

Server

- Sends data to the browser
- Saves information for long term use
- Receives requests from the client

Client

- Receives data from the server
- Renders server data into a usable page
- Handles the user interface

Client and server

Two pieces that talk to each other to make a site

Server

- Sends data to the browser
- Saves information for long term use
- Receives requests from the client

Client

- Receives data from the server
- Renders server data into a usable page
- Handles the user interface

Client and server

Two pieces that talk to each other to make a site

Server

- Sends data to the browser
- Saves information for long term use
- Receives requests from the client

Client

- Receives data from the server
- Renders server data into a usable page
- Handles the user interface

Client and server

Two pieces that talk to each other to make a site

Server

- Sends data to the browser
- Saves information for long term use
- Receives requests from the client

Client

- Receives data from the server
- Renders server data into a usable page
- Handles the user interface

Client and server

Two pieces that talk to each other to make a site

Server

- Sends data to the browser
- Saves information for long term use
- Receives requests from the client

Client

- Receives data from the server
- Renders server data into a usable page
- Handles the user interface

How do you share a site?

- You can load a site locally in your browser
- To share a site you need a server to host
- Free hosting option: `neocities.org`

How do you share a site?

- You can load a site locally in your browser
- To share a site you need a server to host
- Free hosting option: `neocities.org`

How do you share a site?

- You can load a site locally in your browser
- To share a site you need a server to host
- Free hosting option: `neocities.org`

HTML

What does HTML do?

HTML describes the content of the page, but not how it looks

HTML

What does HTML do?

HTML describes the content of the page, but not how it looks

What is HTML?

HyperText Markup Language

- HyperText
- Markup

What is HTML?

HyperText Markup Language

- HyperText
- Markup

HTML in summary

HyperText Markup Language (HTML) is a language that uses *nested tags* to denote what elements a page has and what it should mean

Common tags

Tags we'll need

- `<h1>` ... `<h6>`
- `<ol>`, `<ul>`, `<li>`
- `<p>`
- `<button>`
- `<input>`
- `<script>`
- `<style>`

Common tags

Tags we'll need

- <h1> ... <h6>
- , <ul>,
- <p>
- <button>
- <input>
- <script>
- <style>

Common tags

Tags we'll need

- `<h1> ... <h6>`
- `<ol>`, `<ul>`, `<li>`
- `<p>`
- `<button>`
- `<input>`
- `<script>`
- `<style>`

Common tags

Tags we'll need

- `<h1> ... <h6>`
- `<ol>, <ul>, <li>`
- `<p>`
- `<button>`
- `<input>`
- `<script>`
- `<style>`

Common tags

Tags we'll need

- `<h1> ... <h6>`
- `<ol>, <ul>, <li>`
- `<p>`
- `<button>`
- `<input>`
- `<script>`
- `<style>`

Common tags

Tags we'll need

- `<h1> ... <h6>`
- `<ol>, <ul>, <li>`
- `<p>`
- `<button>`
- `<input>`
- `<script>`
- `<style>`

Common tags

Tags we'll need

- `<h1> ... <h6>`
- `<ol>, <ul>, <li>`
- `<p>`
- `<button>`
- `<input>`
- `<script>`
- `<style>`

Common tags

Tags we'll need

- `<h1> ... <h6>`
- `<ol>, <ul>, <li>`
- `<p>`
- `<button>`
- `<input>`
- `<script>`
- `<style>`

CSS

What does CSS do?

CSS describes how a page looks, but not its content

CSS

What does CSS do?

CSS describes how a page looks, but not its content

CSS properties

Example properties

color	text color
background-color	color of the background
display	visibility and whether elements are displayed as block or inline
font-weight	the boldness of the text

CSS properties

Example properties

color	text color
background-color	color of the background
display	visibility and whether elements are displayed as block or inline
font-weight	the boldness of the text

CSS properties

Example properties

color	text color
background-color	color of the background
display	visibility and whether elements are displayed as block or inline
font-weight	the boldness of the text

CSS properties

Example properties

color	text color
background-color	color of the background
display	visibility and whether elements are displayed as block or inline
font-weight	the boldness of the text

CSS Entries

The shape of CSS

```
selector {
    property: value;
    property: value;
    property: value;
}
```

CSS Entries

The shape of CSS

```
selector {
    property: value;
    property: value;
    property: value;
}
```

CSS Entries

The shape of CSS

```
selector {
    property: value;
    property: value;
    property: value;
}
```

Class

```
.aclass {
    color: red;
    display: inline;
}
```

Class

```
.aclass {
    color: red;
    display: inline;
}
```

Class

```
.aclass {
    color: red;
    display: inline;
}
```

Class

```
.aclass {
    color: red;
    display: inline;
}
```

Id

```
#mybutton {
    font-weight: bold;
}
```

Id

```
#mybutton {
    font-weight: bold;
}
```

Id

```
#mybutton {
    font-weight: bold;
}
```

Tag type

```
p {
    width: 300px;
}
```

Tag type

```
p {
    width: 300px;
}
```

Tag type

```
p {
    width: 300px;
}
```

Exercise 1: Review

Write a simple page that uses

1. At least two CSS selectors
2. At least three different tags

A template

```
<!doctype html>
<html>
  <head>
    <style>
     put your CSS here
    </style>
  </head>
  <body>
   put your HTML here
  </body>
</html>
```

What is the Document Object Model?

The Document Object Model (DOM) gives you the ability to write code that changes the web page dynamically

What is the Document Object Model?

The Document Object Model (DOM) gives you the ability to write code that changes the web page dynamically

What is the Document Object Model?

The Document Object Model (DOM) gives you the ability to write code that changes the web page dynamically

What is the Document Object Model?

The Document Object Model (DOM) gives you the ability to write code that changes the web page dynamically

What does the DOM do?

- Change CSS classes
- Create and remove HTML elements
- Respond to user interface events

What does the DOM do?

- Change CSS classes
- Create and remove HTML elements
- Respond to user interface events

What does the DOM do?

- Change CSS classes
- Create and remove HTML elements
- Respond to user interface events

What does the DOM do?

- Change CSS classes
- Create and remove HTML elements
- Respond to user interface events

What is JavaScript?

JavaScript is a programming language that runs in the browser

What is JavaScript?

JavaScript is a programming language that runs in the browser

What are programming languages?

A programming language is...

- a formal language with rules and grammar
- that has meaning as computation
- and can be used to talk to a computer

What are programming languages?

A programming language is...

- a formal language with rules and grammar
- that has meaning as computation
- and can be used to talk to a computer

What are programming languages?

A programming language is...

- a formal language with rules and grammar
- that has meaning as computation
- and can be used to talk to a computer

Script tag

```
<!doctype html>

<html>
  <head>
    <script>
      ...
    </script>
  </head>
  <body>
    ...
  </body>
</html>
```

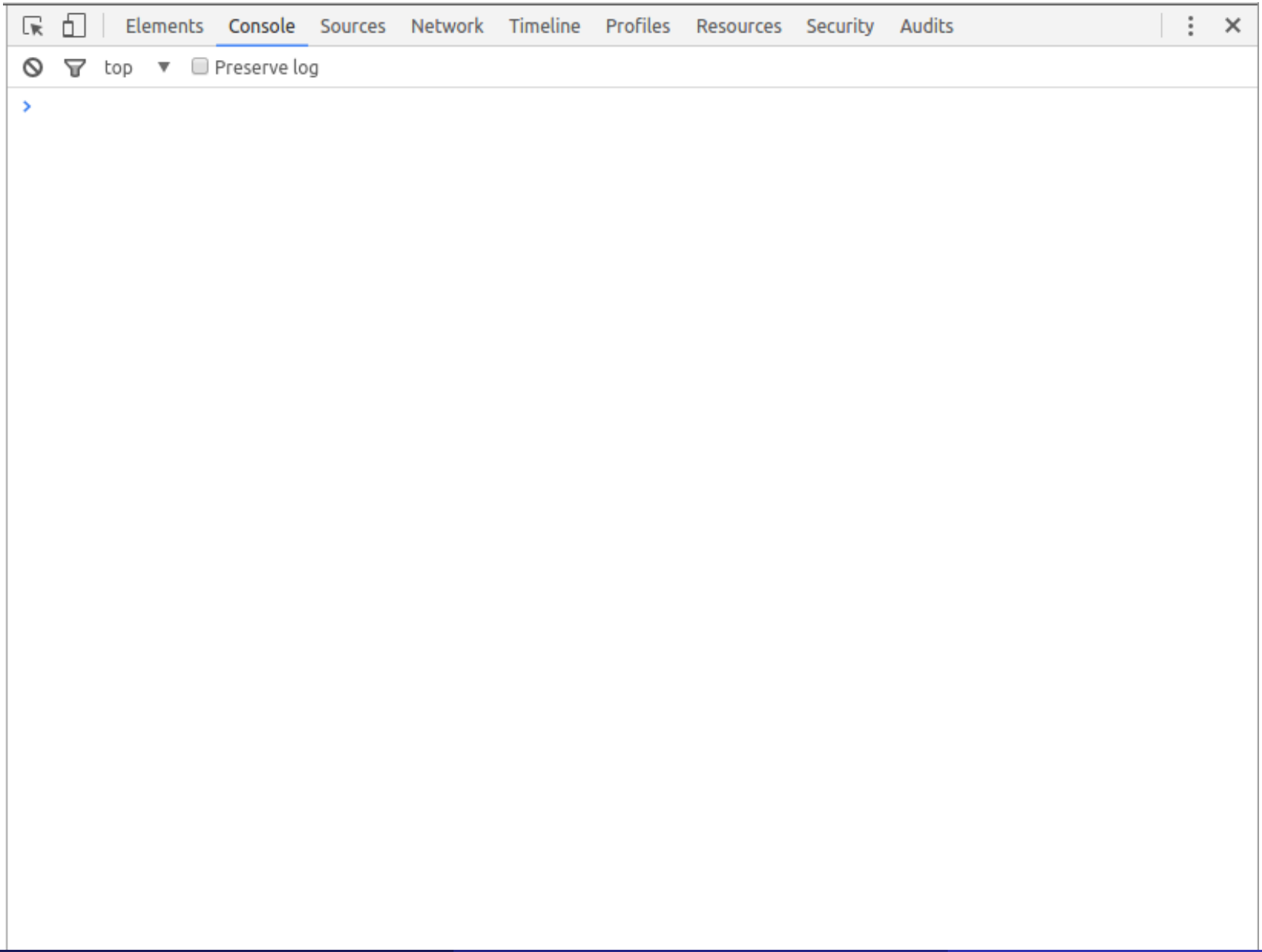
Script tag

```
<!doctype html>

<html>
  <head>
    <script>
      ...
    </script>
  </head>
  <body>
    ...
  </body>
</html>
```

Script tag

```
<!doctype html>

<html>
  <head>
    <script>
      ...
    </script>
  </head>
  <body>
    ...
  </body>
</html>
```

JavaScript console



Basic programming constructs

Data

- Numbers
- Text
- Lists
- Dictionaries

Actions

- Arithmetic
- Creating and using storage
- Performing actions multiple times
- Making choices about what to do
- Naming routine tasks to easily perform them again

Basic programming constructs

Data

- Numbers
- Text
- Lists
- Dictionaries

Actions

- Arithmetic
- Creating and using storage
- Performing actions multiple times
- Making choices about what to do
- Naming routine tasks to easily perform them again

Basic programming constructs

Data

- Numbers
- Text
- Lists
- Dictionaries

Actions

- Arithmetic
- Creating and using storage
- Performing actions multiple times
- Making choices about what to do
- Naming routine tasks to easily perform them again

Basic programming constructs

Data

- Numbers
- Text
- Lists
- Dictionaries

Actions

- Arithmetic
- Creating and using storage
- Performing actions multiple times
- Making choices about what to do
- Naming routine tasks to easily perform them again

Basic programming constructs

Data

- Numbers
- Text
- Lists
- Dictionaries

Actions

- Arithmetic
- Creating and using storage
- Performing actions multiple times
- Making choices about what to do
- Naming routine tasks to easily perform them again

Basic programming constructs

Data

- Numbers
- Text
- Lists
- Dictionaries

Actions

- Arithmetic
- Creating and using storage
- Performing actions multiple times
- Making choices about what to do
- Naming routine tasks to easily perform them again

Basic programming constructs

Data

- Numbers
- Text
- Lists
- Dictionaries

Actions

- Arithmetic
- Creating and using storage
- Performing actions multiple times
- Making choices about what to do
- Naming routine tasks to easily perform them again

Basic programming constructs

Data

- Numbers
- Text
- Lists
- Dictionaries

Actions

- Arithmetic
- Creating and using storage
- Performing actions multiple times
- Making choices about what to do
- Naming routine tasks to easily perform them again

Basic programming constructs

Data

- Numbers
- Text
- Lists
- Dictionaries

Actions

- Arithmetic
- Creating and using storage
- Performing actions multiple times
- Making choices about what to do
- Naming routine tasks to easily perform them again

Basic programming constructs

Data

- Numbers
- Text
- Lists
- Dictionaries

Actions

- Arithmetic
- Creating and using storage
- Performing actions multiple times
- Making choices about what to do
- Naming routine tasks to easily perform them again

Basic programming constructs

Data

- Numbers
- Text
- Lists
- Dictionaries

Actions

- Arithmetic
- Creating and using storage
- Performing actions multiple times
- Making choices about what to do
- Naming routine tasks to easily perform them again

Basic programming constructs

Data

- Numbers
- Text
- Lists
- Dictionaries

Actions

- Arithmetic
- Creating and using storage
- Performing actions multiple times
- Making choices about what to do
- Naming routine tasks to easily perform them again

Strings

Strings are text-as-data, useful for:

- error messages
- writing output

```
"this is a string"
'this is also a string'
"even this 'is a string'"
```

Strings

Strings are text-as-data, useful for:

- error messages
- writing output

```
"this is a string"
'this is also a string'
"even this 'is a string'"
```

Strings

Strings are text-as-data, useful for:

- error messages
- writing output

```
"this is a string"
'this is also a string'
"even this 'is a string'"
```

Strings

Strings are text-as-data, useful for:

- error messages
- writing output

```
"this is a string"
'this is also a string'
"even this 'is a string'"
```

Strings

Strings are text-as-data, useful for:

- error messages
- writing output

```
"this is a string"
'this is also a string'
"even this 'is a string'"
```

Strings

Strings are text-as-data, useful for:

- error messages
- writing output

```
"this is a string"
'this is also a string'
"even this 'is a string'"
```

Strings exercise

Type the following into the console:

- `"hi there everybody"`
- `"it's such a 'nice' day"`
- `'I'm in this class' + ' and I'm typing'`

Variables and Storage

Variables...

- are names given to data
- are storage containers
- can change in value

Type along

```
var thisVariable="a string"
thisVariable
thisVariable = 10
thisVariable
```

Variables and Storage

Variables...

- are names given to data
- are storage containers
- can change in value

Type along

```
var thisVariable="a string"
thisVariable
thisVariable = 10
thisVariable
```

Variables and Storage

Variables...

- are names given to data
- are storage containers
- can change in value

Type along

```
var thisVariable="a string"
thisVariable
thisVariable = 10
thisVariable
```

Variables and Storage

Variables...

- are names given to data
- are storage containers
- can change in value

Type along

```
var thisVariable="a string"
thisVariable
thisVariable = 10
thisVariable
```

Variables and Storage

Variables...

- are names given to data
- are storage containers
- can change in value

Type along

```
var thisVariable="a string"
thisVariable
thisVariable = 10
thisVariable
```

Variables and Storage

Variables...

- are names given to data
- are storage containers
- can change in value

Type along

```
var thisVariable="a string"
thisVariable
thisVariable = 10
thisVariable
```

Variables and Storage

Variables...

- are names given to data
- are storage containers
- can change in value

Type along

```
var thisVariable="a string"
thisVariable
thisVariable = 10
thisVariable
```

Variables and Storage

Variables...

- are names given to data
- are storage containers
- can change in value

Type along

```
var thisVariable="a string"
thisVariable
thisVariable = 10
thisVariable
```

Lists and Arrays

In JavaScript, the data type for lists are called arrays

Arrays

- have a beginning and end
- are in order
- can be accessed by index

Type along

```
var myArr = [1,2,3]
myArr
myArr[0]
myArr[1]
myArr[2]
myArr[0] = 20
myArr[0]
```

Lists and Arrays

In JavaScript, the data type for lists are called arrays

Arrays

- have a beginning and end
- are in order
- can be accessed by index

Type along

```
var myArr = [1,2,3]
myArr
myArr[0]
myArr[1]
myArr[2]
myArr[0] = 20
myArr[0]
```

Lists and Arrays

In JavaScript, the data type for lists are called arrays

Arrays

- have a beginning and end
- are in order
- can be accessed by index

Type along

```
var myArr = [1,2,3]
myArr
myArr[0]
myArr[1]
myArr[2]
myArr[0] = 20
myArr[0]
```

Lists and Arrays

In JavaScript, the data type for lists are called arrays

Arrays

- have a beginning and end
- are in order
- can be accessed by index

Type along

```
var myArr = [1,2,3]
myArr
myArr[0]
myArr[1]
myArr[2]
myArr[0] = 20
myArr[0]
```

Lists and Arrays

In JavaScript, the data type for lists are called arrays

Arrays

- have a beginning and end
- are in order
- can be accessed by index

Type along

```
var myArr = [1,2,3]
myArr
myArr[0]
myArr[1]
myArr[2]
myArr[0] = 20
myArr[0]
```

Lists and Arrays

In JavaScript, the data type for lists are called arrays

Arrays

- have a beginning and end
- are in order
- can be accessed by index

Type along

```
var myArr = [1,2,3]
myArr
myArr[0]
myArr[1]
myArr[2]
myArr[0] = 20
myArr[0]
```

Lists and Arrays

In JavaScript, the data type for lists are called arrays

Arrays

- have a beginning and end
- are in order
- can be accessed by index

Type along

```
var myArr = [1,2,3]
myArr
myArr[0]
myArr[1]
myArr[2]
myArr[0] = 20
myArr[0]
```

Lists and Arrays

In JavaScript, the data type for lists are called arrays

Arrays

- have a beginning and end
- are in order
- can be accessed by index

Type along

```
var myArr = [1,2,3]
myArr
myArr[0]
myArr[1]
myArr[2]
myArr[0] = 20
myArr[0]
```

Lists and Arrays

In JavaScript, the data type for lists are called arrays

Arrays

- have a beginning and end
- are in order
- can be accessed by index

Type along

```
var myArr = [1,2,3]
myArr
myArr[0]
myArr[1]
myArr[2]
myArr[0] = 20
myArr[0]
```

Lists and Arrays

In JavaScript, the data type for lists are called arrays

Arrays

- have a beginning and end
- are in order
- can be accessed by index

Type along

```
var myArr = [1,2,3]
myArr
myArr[0]
myArr[1]
myArr[2]
myArr[0] = 20
myArr[0]
```

Lists and Arrays

In JavaScript, the data type for lists are called arrays

Arrays

- have a beginning and end
- are in order
- can be accessed by index

Type along

```
var myArr = [1,2,3]
myArr
myArr[0]
myArr[1]
myArr[2]
myArr[0] = 20
myArr[0]
```

Objects and Dictionaries

Objects...

- are like dictionaries
- associate names and data
- are used to collect information

Type along

```
var obj = {names : "chicken",
           petSpecies : "dog",
           age : 10}
obj.names
obj.petSpecies
obj.age = 10
obj.age
```

Objects and Dictionaries

Objects...

- are like dictionaries
- associate names and data
- are used to collect information

Type along

```
var obj = {names : "chicken",
           petSpecies : "dog",
           age : 10}
obj.names
obj.petSpecies
obj.age = 10
obj.age
```

Objects and Dictionaries

Objects...

- are like dictionaries
- associate names and data
- are used to collect information

Type along

```
var obj = {names : "chicken",
           petSpecies : "dog",
           age : 10}
obj.names
obj.petSpecies
obj.age = 10
obj.age
```

Objects and Dictionaries

Objects...

- are like dictionaries
- associate names and data
- are used to collect information

Type along

```
var obj = {names : "chicken",
           petSpecies : "dog",
           age : 10}
obj.names
obj.petSpecies
obj.age = 10
obj.age
```

Objects and Dictionaries

Objects...

- are like dictionaries
- associate names and data
- are used to collect information

Type along

```
var obj = {names : "chicken",
           petSpecies : "dog",
           age : 10}
obj.names
obj.petSpecies
obj.age = 10
obj.age
```

Objects and Dictionaries

Objects...

- are like dictionaries
- associate names and data
- are used to collect information

Type along

```
var obj = {names : "chicken",
           petSpecies : "dog",
           age : 10}
obj.names
obj.petSpecies
obj.age = 10
obj.age
```

Objects and Dictionaries

Objects...

- are like dictionaries
- associate names and data
- are used to collect information

Type along

```
var obj = {names : "chicken",
           petSpecies : "dog",
           age : 10}
obj.names
obj.petSpecies
obj.age = 10
obj.age
```

Objects and Dictionaries

Objects...

- are like dictionaries
- associate names and data
- are used to collect information

Type along

```
var obj = {names : "chicken",
           petSpecies : "dog",
           age : 10}
obj.names
obj.petSpecies
obj.age = 10
obj.age
```

Objects and Dictionaries

Objects...

- are like dictionaries
- associate names and data
- are used to collect information

Type along

```
var obj = {names : "chicken",
           petSpecies : "dog",
           age : 10}
obj.names
obj.petSpecies
obj.age = 10
obj.age
```

Functions

Functions...

- are code that can be used again and again
- are data that can be assigned to variables
- use the keyword `return` to give back a value
- are made up of arguments and a body

```
function () {
//this function doesn't have a name
   var x = 10;
   return x + x;
}

function thisFun (x,y) {
//but this function does
   return (x + y);
}
```

Functions

Functions...

- are code that can be used again and again
- are data that can be assigned to variables
- use the keyword return to give back a value
- are made up of arguments and a body

```
function () {
//this function doesn't have a name
   var x = 10;
   return x + x;
}

function thisFun (x,y) {
//but this function does
   return (x + y);
}
```

Functions

Functions...

- are code that can be used again and again
- are data that can be assigned to variables
- use the keyword `return` to give back a value
- are made up of arguments and a body

```
function () {
//this function doesn't have a name
   var x = 10;
   return x + x;
}

function thisFun (x,y) {
//but this function does
   return (x + y);
}
```

Functions

Functions...

- are code that can be used again and again
- are data that can be assigned to variables
- use the keyword `return` to give back a value
- are made up of arguments and a body

```
function () {
//this function doesn't have a name
   var x = 10;
   return x + x;
}

function thisFun (x,y) {
//but this function does
   return (x + y);
}
```

Functions

Functions...

- are code that can be used again and again
- are data that can be assigned to variables
- use the keyword `return` to give back a value
- are made up of arguments and a body

```
function () {
//this function doesn't have a name
   var x = 10;
   return x + x;
}

function thisFun (x,y) {
//but this function does
   return (x + y);
}
```

Functions

Functions...

- are code that can be used again and again
- are data that can be assigned to variables
- use the keyword `return` to give back a value
- are made up of arguments and a body

```
function () {
//this function doesn't have a name
   var x = 10;
   return x + x;
}

function thisFun (x,y) {
//but this function does
   return (x + y);
}
```

Functions

Functions...

- are code that can be used again and again
- are data that can be assigned to variables
- use the keyword `return` to give back a value
- are made up of arguments and a body

```
function () {
//this function doesn't have a name
   var x = 10;
   return x + x;
}

function thisFun (x,y) {
//but this function does
   return (x + y);
}
```

Functions

Functions...

- are code that can be used again and again
- are data that can be assigned to variables
- use the keyword `return` to give back a value
- are made up of arguments and a body

```
function () {
//this function doesn't have a name
    var x = 10;
    return x + x;
}

function thisFun (x,y) {
//but this function does
    return (x + y);
}
```

Functions

Functions...

- are code that can be used again and again
- are data that can be assigned to variables
- use the keyword `return` to give back a value
- are made up of arguments and a body

```
function () {
//this function doesn't have a name
   var x = 10;
   return x + x;
}

function thisFun (x,y) {
//but this function does
   return (x + y);
}
```

Functions

Functions...

- are code that can be used again and again
- are data that can be assigned to variables
- use the keyword `return` to give back a value
- are made up of arguments and a body

```
function () {
//this function doesn't have a name
   var x = 10;
   return x + x;
}

function thisFun (x,y) {
//but this function does
   return (x + y);
}
```

Functions

Functions...

- are code that can be used again and again
- are data that can be assigned to variables
- use the keyword `return` to give back a value
- are made up of arguments and a body

```
function () {
//this function doesn't have a name
   var x = 10;
   return x + x;
}

function thisFun (x,y) {
//but this function does
   return (x + y);
}
```

Functions

Functions...

- are code that can be used again and again
- are data that can be assigned to variables
- use the keyword `return` to give back a value
- are made up of arguments and a body

```
function () {
//this function doesn't have a name
   var x = 10;
   return x + x;
}

function thisFun (x,y) {
//but this function does
   return (x + y);
}
```

How we'll proceed

From here on, we'll be presenting examples of JavaScript interacting with the DOM and practice more JavaScript from there

How we'll proceed

From here on, we'll be presenting examples of JavaScript interacting with the DOM and practice more JavaScript from there

What is the Document Object Model?

The DOM

The document object model (DOM) is the representation of the web page *as JavaScript objects*

Putting the document in DOM

The document object

`document` is the object that holds most of the important methods for controlling web pages

Our first example

In our first example we'll

- create an HTML element in JavaScript
- create text to put inside the element
- insert the HTML element in the web page

Our first example

In our first example we'll

- create an HTML element in JavaScript
- create text to put inside the element
- insert the HTML element in the web page

Our first example

In our first example we'll

- create an HTML element in JavaScript
- create text to put inside the element
- insert the HTML element in the web page

Our first example

In our first example we'll

- create an HTML element in JavaScript
- create text to put inside the element
- insert the HTML element in the web page

When to load code

```
window.onload = function () {
    ...
};
```

Creating elements

Relevant functions

- `document.createElement`
- `document.createTextNode`
- `element.appendChild`
- `document.body`

Creating elements

```
<!doctype html>

<html>
  <head>
    <script>
      window.onload = function () {
        var elm = document.createElement("p");
        var text = document.createTextNode("this is text");
        elm.appendChild(text);
        document.body.appendChild(elm);
      }
    </script>
  </head>
  <body>
  </body>
</html>
```

Creating elements

```
<!doctype html>

<html>
  <head>
    <script>
      window.onload = function () {
        var elm = document.createElement("p");
        var text = document.createTextNode("this is text");
        elm.appendChild(text);
        document.body.appendChild(elm);
      }
    </script>
  </head>
  <body>
  </body>
</html>
```

Creating elements

```
<!doctype html>

<html>
  <head>
    <script>
      window.onload = function () {
        var elm = document.createElement("p");
        var text = document.createTextNode("this is text");
        elm.appendChild(text);
        document.body.appendChild(elm);
      }
    </script>
  </head>
  <body>
  </body>
</html>
```

Creating elements

```
<!doctype html>

<html>
  <head>
    <script>
      window.onload = function () {
        var elm = document.createElement("p");
        var text = document.createTextNode("this is text");
        elm.appendChild(text);
        document.body.appendChild(elm);
      }
    </script>
  </head>
  <body>
  </body>
</html>
```

Creating elements

```
<!doctype html>

<html>
  <head>
    <script>
      window.onload = function () {
        var elm = document.createElement("p");
        var text = document.createTextNode("this is text");
        elm.appendChild(text);
        document.body.appendChild(elm);
      }
    </script>
  </head>
  <body>
  </body>
</html>
```

Creating elements

```
<!doctype html>

<html>
  <head>
    <script>
      window.onload = function () {
        var elm = document.createElement("p");
        var text = document.createTextNode("this is text");
        elm.appendChild(text);
        document.body.appendChild(elm);
      }
    </script>
  </head>
  <body>
  </body>
</html>
```

Creating elements

```
<!doctype html>

<html>
  <head>
    <script>
      window.onload = function () {
        var elm = document.createElement("p");
        var text = document.createTextNode("this is text");
        elm.appendChild(text);
        document.body.appendChild(elm);
      }
    </script>
  </head>
  <body>
  </body>
</html>
```

Your turn

- Create a new html file
- Leave the body empty
- Create two elements and put them in the body

```
<!doctype html>
<html>
  <head>
    <script>
      window.onload = function () {
        ...
      }
    </script>
  </head>
  <body>
  </body>
</html>
```

Your turn

- Create a new html file
- Leave the body empty
- Create two elements and put them in the body

```
<!doctype html>
<html>
  <head>
    <script>
      window.onload = function () {
        ...
      }
    </script>
  </head>
  <body>
  </body>
</html>
```

Getting existing elements

- `document.getElementById`
- `document.getElementsByTagName`
- `element.firstChild`
- `node.nodeValue`

Getting existing elements

- `document.getElementById`
- `document.getElementsByTagName`
- `element.firstChild`
- `node.nodeValue`

Getting existing elements

- `document.getElementById`
- `document.getElementsByTagName`
- `element.firstChild`
- `node.nodeValue`

Getting existing elements

- `document.getElementById`
- `document.getElementsByTagName`
- `element.firstChild`
- `node.nodeValue`

getElementById

```
<body>
  <ol id="list1">
    <li>This is a list</li>
  </ol>
  <ol id="list2">
    <li>This is our second list</li>
  </ol>
</body>
```

getElementById

```
window.onload = function () {
    var newItem =
      document.createElement("li");
    var newText =
        document
        .createTextNode("item in the second list");
    newItem.appendChild(newText);
    var secondList = document.getElementById("list2");
    secondList.appendChild(newItem);
};
```

Your turn

- Create a new html file
- Follow the template to the right
- Add an element to the list

```
<!doctype html>
<html>
  <head>
    <script>
      window.onload = function () {
        ...
      }
    </script>
  </head>
  <body>
    <ol id="list">
    </ol>
  </body>
</html>
```

Changing CSS properties

Important properties and methods

- `elm.style`
- `elm.classList`
- `elm.classList.add`
- `elm.classList.remove`

Changing CSS properties

Important properties and methods

- `elm.style`
- `elm.classList`
- `elm.classList.add`
- `elm.classList.remove`

Changing CSS properties

Important properties and methods

- `elm.style`
- `elm.classList`
- `elm.classList.add`
- `elm.classList.remove`

Changing CSS properties

Important properties and methods

- `elm.style`
- `elm.classList`
- `elm.classList.add`
- `elm.classList.remove`

Changing CSS properties

Important properties and methods

- `elm.style`
- `elm.classList`
- `elm.classList.add`
- `elm.classList.remove`

Changing CSS properties

```
<!doctype html>
<html>
  <head>
    <script>
      window.onload = function () {
        var h = document.getElementById("heading");
        h.style.color = "red";
      }
    </script>
  </head>
  <body>
    <h1 id="heading">This is a heading!</h1>
  </body>
</html>
```

Changing the CSS class

```
<head>
  <style>
    .reddish {
      color: red;
    }
  </style>
  <script>
    window.onload = function () {
       var h = document.getElementById("heading");
       h.classList.add("reddish");
    };
  </script>
</head>
```

Events

What are events?

Events connect user interfaces to code

- Mouse clicks
- Keys pressed
- Moving your cursor
- Focusing on an element

Events

What are events?

Events connect user interfaces to code

- Mouse clicks
- Keys pressed
- Moving your cursor
- Focusing on an element

Events

What are events?

Events connect user interfaces to code

- Mouse clicks
- Keys pressed
- Moving your cursor
- Focusing on an element

Events

What are events?

Events connect user interfaces to code

- Mouse clicks
- Keys pressed
- Moving your cursor
- Focusing on an element

Events

What are events?

Events connect user interfaces to code

- Mouse clicks
- Keys pressed
- Moving your cursor
- Focusing on an element

Events

Listening for events

- `elem.addEventListener`
- `elem.removeEventListener`

Events

Listening for events

- `elem.addEventListener`
- `elem.removeEventListener`

Events

Listening for events

- `elem.addEventListener`
- `elem.removeEventListener`

Listening to events

```
<head>
  <script>
    window.onload = function () {
       var h = document.getElementById("heading");
       h.addEventListener("mouseover", function () {
          this.style.color = "red";
       });
       h.addEventListener("mouseleave", function () {
          this.style.color = "black";
       });
    };
  </script>
</head>
<body>
  <h1 id="heading">This is our heading!</h1>
</body>
```

Collapsing list

```
<body>
  <div id="content">
    <h3>Our list is below here</h3>
    <ol id="list">
      <li>First item</li>
      <li>Second item</li>
      <li>Third item</li>
      <li>Fourth item</li>
    </ol>
  </div>
</body>
```

Collapsing list

```
window.onload = function () {
    var list = document.getElementById("list");
    var div = document.getElementById("content");
    div.addEventListener("mouseover", function () {
        list.style.display = "block";
    });
    div.addEventListener("mouseleave", function () {
        list.style.display = "none";
    });
};
```

To-do list

```
<body>
  <h1>Welcome to your to-do list</h1>
  <ol id="list">
  </ol>
  <input id="input" type="text"></input>
  <button id="add">Add element</button>
</body>
```

To-do list

```
var inputElement = document.getElementById("input");
var todoList = document.getElementById("list");
var addButton = document.getElementById("add");

addButton.addEventListener("click", function () {
  var itemText = document.createTextNode(inputElement.value);
  var newItem = document.createElement("li");
  newItem.appendChild(itemText);
  todoList.appendChild(newItem);
  inputElement.value = "";
});
```

To-do list

```
inputElement.addEventListener("focus", function () {
    inputElement.style.fontWeight = "bold";
});

inputElement.addEventListener("blur", function () {
    inputElement.style.fontWeight = "normal";
});
```

What's left

- A lot more JavaScript
- Frameworks
- Servers
- Experimenting

What's left

- A lot more JavaScript
- Frameworks
- Servers
- Experimenting

What's left

- A lot more JavaScript
- Frameworks
- Servers
- Experimenting

What's left

- A lot more JavaScript
- Frameworks
- Servers
- Experimenting