GENERAL ASSEMBLY

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Pull changes from the `svodnik/jsd6` repo to your computer
2. Open the `starter-code` folder in your code editor

JAVASCRIPT DEVELOPMENT

THE MODULE PATTERN & THIS

LEARNING OBJECTIVES

At the end of this class, you will be able to

- Implement the module pattern in your code.
- Understand and explain Javascript context.

AGENDA

- The module pattern
- Context and `this`

Prototypal Inheritance — Review

- Think about the relationship between a constructor function, the `prototype` property, and an object created from that constructor function
- On your desk, draw a diagram or write a description that explains how these three things are related.

Checkin and questions

- The **most significant thing I learned** about prototypal inheritance is _____.
- My **biggest outstanding question** about prototypal inheritance is _____.

Exit Tickets – Prototypes

- Practical applications. “When would I use this while building my friend’s webpage?”
- “I think this lesson would make sense after the object lesson and before DOM”
- Will we be getting homework feedback for the last few weeks?
- “I have a feeling we might have missed something”

How do you know what someone's talking about when they don't specifically name the thing?

CLOSURES - REVIEW

- A **closure** is an inner function that has access to the outer (enclosing) function's variables.
- You create a closure by adding a function inside another function.
- A closure is also known as **lexical scope**

CLOSURES — KEY POINTS

- Closures have access to the outer function's variables (including parameters) **even after the outer function returns**.
- Closures store **references** to the outer function's variables.

THE MODULE PATTERN

- Using an IIFE to return an object literal
- The methods of the returned object can access the private properties and methods of the IIFE (closures!), but other code cannot do this
- This means specific parts of the IIFE are not available in the global scope

BENEFITS OF THE MODULE PATTERN

- Lets you keep some functions and variables private
- Lets you avoid polluting the global scope
- Keeps code organized into objects

CONTEXT AND THIS

- Functions do not occur in a vacuum - they are always executed in relation to some object
- **Context** refers to whatever object is responsible for executing a function
- This object can be referenced using the keyword `this`
- In other words, `this` represents whatever object is in context when a function runs

CONTEXT RULES

situation	what `this` maps to
function invocation	default: the global object (`window`) strict mode: `undefined`
method invocation	the object that owns the method
constructor function	the newly created object
event handler	the element that the event was fired from

MANIPULATING CONTEXT

There are three methods that allow us to control context:

- `call`: Calls a function with a given `this` value and arguments provided individually
- `apply`: Calls a function with a given `this` value and arguments provided as an array (or an array-like object)
- `bind`: Creates a new function that, when called, has its `this` keyword set to the provided value, with a given sequence of arguments preceding any provided when the new function is called

MANIPULATING CONTEXT WITH CALL()

```
var user = {
  firstName: 'Barack',
  lastName: 'Obama',
  showFullName: function() {
   console.log(this.firstName + ' ' + this.lastName)
  }
}

$(‘.button').click(function() {
  user.showFullName.call(user) // Barack Obama
})
```

MANIPULATING CONTEXT WITH APPLY()

```
var user = {
  firstName: 'Barack',
  lastName: 'Obama',
  showFullName: function() {
   console.log(this.firstName + ' ' + this.lastName)
  }
}

$('.button').click(function() {
  user.showFullName.apply(user) // Barack Obama
})
```

MANIPULATING CONTEXT WITH BIND()

```
  // declare a new variable whose value is the
user.showFullName function with a context set to user
  var contextSetUser = user.showFullName.bind(user);
  $(‘button').click(contextSetUser);
  // Barack Obama

  $(‘button').click(user.showFullName);
  // undefined undefined
```

LEARNING OBJECTIVES - REVIEW

- Implement the module pattern in your code.
- Understand and explain Javascript context.

NEXT CLASS PREVIEW

In-class lab: Intro to CRUD and Firebase

- Explain what CRUD is. (**Preview**: Create, Read, Update, Delete)
- Explain the HTTP methods associated with CRUD.
- Implement Firebase in an application.
- Build a full-stack app.

Exit Tickets!

Q&A