Javascript intro

Jonathan Geisler

February 12, 2007

What's in a name?

- Livescript (Netscape)

What's in a name?

- Livescript (Netscape)
- *Javascript* (Netscape & Sun partnership)

What's in a name?

- Livescript (Netscape)
- *Javascript* (Netscape & Sun partnership)
  - Sun involved, so Java name included

What's in a name?

- Livescript (Netscape)
- *Javascript* (Netscape & Sun partnership)
  - Sun involved, so Java name included
  - NOT much like Java

What's in a name?

- Livescript (Netscape)
- *Javascript* (Netscape & Sun partnership)
  - Sun involved, so Java name included
  - NOT much like Java
- ECMAscript

What's in a name?

- Livescript (Netscape)
- *Javascript* (Netscape & Sun partnership)
  - Sun involved, so Java name included
  - NOT much like Java
- ECMAscript
  - IE & Mozilla/Netscape interoperability

What's in a name?

- Livescript (Netscape)
- *Javascript* (Netscape & Sun partnership)
  - Sun involved, so Java name included
  - NOT much like Java
- ECMAscript
  - IE & Mozilla/Netscape interoperability
  - Better name, but very few know what it is

Where is Javascript used?

- *Client side scripts*

Where is Javascript used?

- *Client side scripts*
- Server side scripts

Where is Javascript used?

- *Client side scripts*
- Server side scripts
- Stand-alone applications

Types

- Dynamically

Types

- Dynamically
  - Created

Types

- Dynamically
  - Created
  - Checked

Types

- Dynamically
  - Created
  - Checked
  - Referenced

Types

- Dynamically
  - Created
  - Checked
  - Referenced
- Everything is like an

Types

- Dynamically
  - Created
  - Checked
  - Referenced
- Everything is like an
  - Object

Types

- Dynamically
  - Created
  - Checked
  - Referenced
- Everything is like an
  - Object
  - Array

Syntax

OK, there is *one* Java-like thing in Javascript . . . Most everything looks like C++, except:

- No `classes`

Syntax

OK, there is *one* Java-like thing in Javascript . . . Most everything looks like C++, except:

- No `classes`
- `typeof`

Syntax

OK, there is *one* Java-like thing in Javascript . . . Most everything looks like C++, except:

- No `classes`
- `typeof`
- `for-in`

Syntax

OK, there is *one* Java-like thing in Javascript . . . Most everything looks like C++, except:

- No `classes`
- `typeof`
- `for-in`
- Exceptions more Java-like with `finally` clause

Syntax

OK, there is *one* Java-like thing in Javascript . . . Most everything looks like C++, except:

- No `classes`
- `typeof`
- `for-in`
- Exceptions more Java-like with `finally` clause
- `var` to declare variables

Primitives

Javascript has these things called primitives, but due to coercions, you normally don't need to worry about them. They are closer to machine representation of types like integers or floating point numbers.

Operators

- C++ and Java operators are standard

Operators

- C++ and Java operators are standard
- ===[1] and !== are new and prevent coercions

[1] Can you remember that many equal signs?

Operators

- C++ and Java operators are standard
- `===`[1] and `!==` are new and prevent coercions
- `typeof` returns a string representation of a variable's type, but not very helpful since there are so few types

[1] Can you remember that many equal signs?

Properties

- Data & Method types

Properties

- Data & Method types
- Dynamically created & deleted

Properties

- Data & Method types
- Dynamically created & deleted
- Accessed with `.` & `[]`

For use with browsers

Javascript was developed for browsing the web, so it has functions that interact with the browser user:

- `alert()`
- `confirm()`
- `prompt()`
- `document.write()`

Does Javascript have classes?

- Yes

Does Javascript have classes?

- Yes
  - `new` can be called with any function as a constructor

Does Javascript have classes?

- Yes
  - `new` can be called with any function as a constructor
  - Objects have both data and methods

Does Javascript have classes?

- Yes
  - `new` can be called with any function as a constructor
  - Objects have both data and methods
- No

Does Javascript have classes?

- Yes
  - `new` can be called with any function as a constructor
  - Objects have both data and methods
- No
  - No Inheritance

Does Javascript have classes?

- Yes
  - `new` can be called with any function as a constructor
  - Objects have both data and methods
- No
  - No Inheritance
  - No Polymorphism

Does Javascript have classes?

- Yes
  - `new` can be called with any function as a constructor
  - Objects have both data and methods
- No
  - No Inheritance
  - No Polymorphism
  - Severely limited Encapsulation

Does Javascript have classes?

- Yes
  - `new` can be called with any function as a constructor
  - Objects have both data and methods
- No
  - No Inheritance
  - No Polymorphism
  - Severely limited Encapsulation
  - No type association with a class name

Everything is an array?

- Yes!

Everything is an array?

- Yes!
- Use the `Array()` constructor, or
- Use the `[]` constructor

Everything is an array?

- Yes!
- Use the `Array()` constructor, or
- Use the `[]` constructor
- Use `for-in` to get at all the elements

Everything is an array?

- Yes!
- Use the `Array()` constructor, or
- Use the `[]` constructor
- Use `for-in` to get at all the elements
- Use `.length` to get/set the last element location

Everything is an array?

- Yes!
- Use the `Array()` constructor, or
- Use the `[]` constructor
- Use `for-in` to get at all the elements
- Use `.length` to get/set the last element location
- Perl-like operators available (e.g., `push()`, `pop()`, `shift()`, `unshift()`, etc.)

Variable scope

Everything is global except for locally defined variables (`var`) inside a function. Therefore, use functions for name safety.

Regular Expressions

Look forward to Perl, where we'll spend time on this explicitly! When that happens, note these similarites:

- `search()` or `match()` in Javascript is like `m//` in Perl
- `replace()` in Javascript is like `s///` in Perl
- Patterns/classes in Javascript are nearly identical to Perl (intentionally)

Event based programming

- We will see this in Java, too
- Response to user allows better *interactivity*
- Independent code for each action
- Flow of control is implicit

Calling Javascript from HTML

- Use `<script>` explicitly and do one of the following:
  1. Have source inside tag
  2. Use `src` attribute to point to separate source file
- Include handlers for desired events with attributes on desired tags (e.g., `<body onload="dostuff()">`

Handlers

The book does a great job on pages 197–199 of listing all the Javascript events. Some of the most frequently used events are:

- `onclick`
- `onsubmit/onreset`
- `onload`
- `onmouseover/onmouseout`

What can be done with Javascript?

- Modify the HTML document directly
- Form validation (often done)
- Snazzy presentation
  - Using mouseover
  - Using clicks

Squirrelmail is a great example! And it all started here at Taylor!

Document Object Model (DOM) history

1. Javascript for Netscape (DOM 0)

Document Object Model (DOM) history

1. Javascript for Netscape (DOM 0)
2. IE and others copied it for compatibility purposes

Document Object Model (DOM) history

1. Javascript for Netscape (DOM 0)
2. IE and others copied it for compatibility purposes
3. IE & Netscape extended the DOM in different ways to support DHTML

Document Object Model (DOM) history

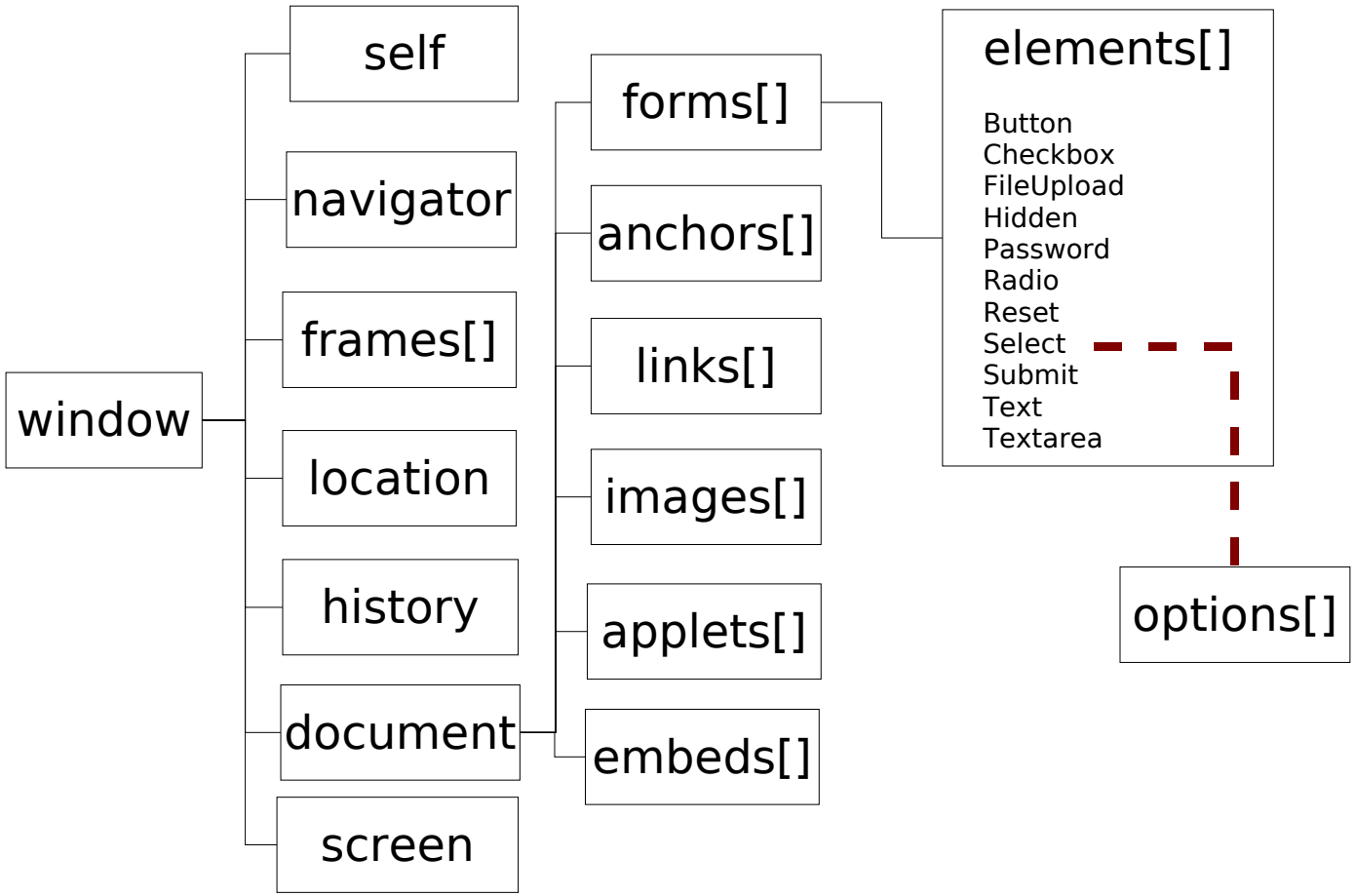
1. Javascript for Netscape (DOM 0)
2. IE and others copied it for compatibility purposes
3. IE & Netscape extended the DOM in different ways to support DHTML
4. W3C standardized two versions (DOM 1 & 2) with support from Microsoft and Netscape, but only Netscape tried to fully comply

Document Object Model (DOM) history

1. Javascript for Netscape (DOM 0)
2. IE and others copied it for compatibility purposes
3. IE & Netscape extended the DOM in different ways to support DHTML
4. W3C standardized two versions (DOM 1 & 2) with support from Microsoft and Netscape, but only Netscape tried to fully comply

Therefore, still use DOM 0 if at all possible. This may change in the future as Microsoft just released IE 7 beta this summer. The expectation is that it will have better standards compliance (including DOM 2) and that you may be able to start using those advanced features. We're going to move more towards using DOM 2 with AJAX, but doing it intelligently so that browsers that don't support it are still able to display our pages and function as good as possible given their limitations.

DOM Level 0 hierarchy



The model views the page (document) as a tree. You may traverse the tree by:

- Explicitly visiting each node until you find the one you're interested via the `childNodes[]`, `parentNode`, `firstChild`, and/or `lastChild` properties.
- Let the browser find the node you're interested in automatically via `getElementByID()` or `getElementsByTag()`

What can we do with DOM nodes?

1. Change attributes

What can we do with DOM nodes?

1. Change attributes
2. Change text

What can we do with DOM nodes?

1. Change attributes
2. Change text
3. Change styles

What can we do with DOM nodes?

1. Change attributes
2. Change text
3. Change styles
4. Move them to a different spot in the document

What can we do with DOM nodes?

1. Change attributes
2. Change text
3. Change styles
4. Move them to a different spot in the document
5. Delete them from the document

What can we do with DOM nodes?

1. Change attributes
2. Change text
3. Change styles
4. Move them to a different spot in the document
5. Delete them from the document
6. Create new nodes for the document

What can we do with DOM nodes?

1. Change attributes
2. Change text
3. Change styles
4. Move them to a different spot in the document
5. Delete them from the document
6. Create new nodes for the document
7. "Clone" them (i.e., make a copy)

Author's examples

The author gives two examples:

1. Expanding a form to fit the number of entries exactly without needing extra entries. Think of this as the web equivalent of `new` because we are able to use exactly the right amount of memory without having declare too much to be sure we have enough for any plausible situation.
2. Generic rollovers. Basically you have one piece of code that should work on any web page that includes this script and then declares the `class` for the `<img>` tag to be a rollover.

What thoughts and questions did you have about these examples?

Connecting functions to events

As your book points out, there is the DOM way, and the IE way. We'll look at the DOM to simplfy things and let you get the differences for IE from the book.
To have `myClicker()` called when we click on the element identified by `clicker`, we use the following code:

```
function myClicker(e) {
    // code
}

var elem = document.getElementById("clicker");
elem.addEventListener("click", myClicker, false);
```

Changing the normal browser behavior

In order to stop an event from being sent to each element up the DOM tree, we use `stopPropagation()`.

In order to prevent an event from causing default behavior, we use `preventDefault()`.

Smart Events

We'll cover the examples in your book on the board. What questions do you have concerning the "Creating Smarter Links" and "Making Tables More Readable" sections?