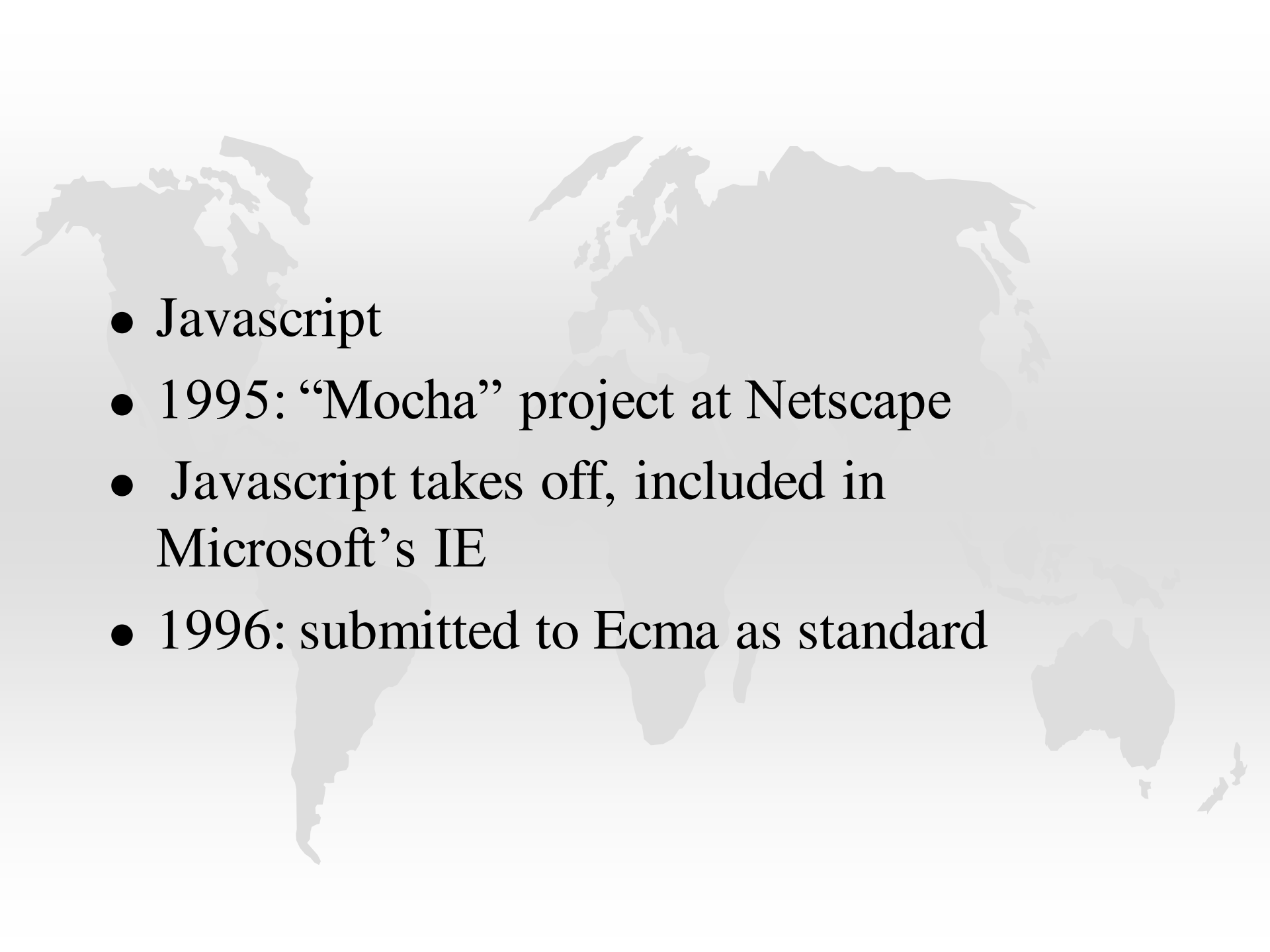


A light gray world map is centered in the background of the slide. The map shows the outlines of the continents: North America, South America, Europe, Africa, Asia, and Australia. The text "JavaScript" is overlaid on the map, positioned over the Atlantic Ocean and parts of Europe and Africa.

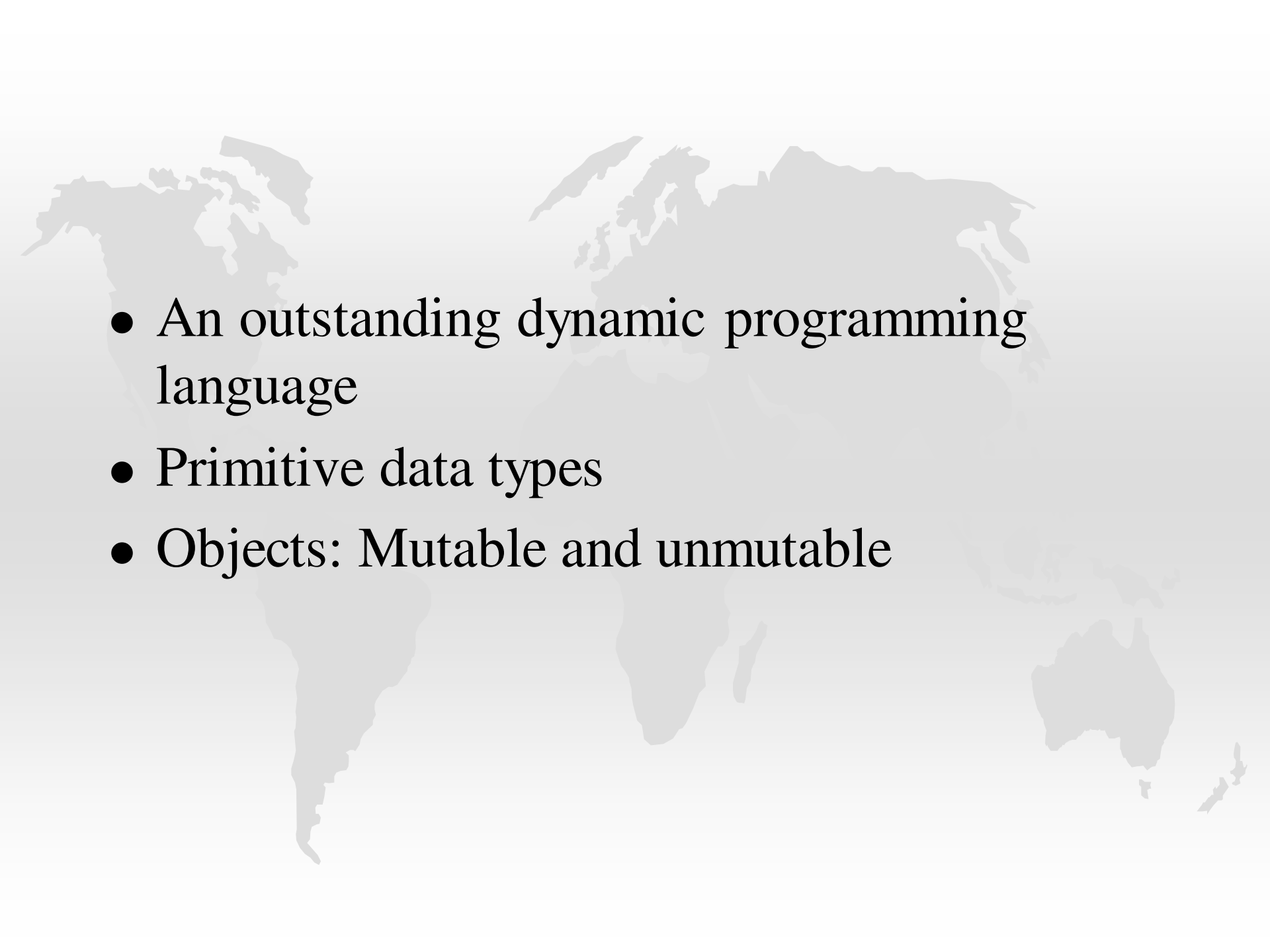
JavaScript

- 
- A faint, light gray world map is visible in the background of the slide, showing the continents of North America, South America, Europe, Africa, Asia, and Australia.
- Javascript
 - 1995: “Mocha” project at Netscape
 - Javascript takes off, included in Microsoft’s IE
 - 1996: submitted to Ecma as standard

Today



- Java alive and well server-side
- but JS dominates client-side
- making inroads server-side too (eg, node.js)

- 
- An outstanding dynamic programming language
 - Primitive data types
 - Objects: Mutable and immutable

syntax

- statements like Java
- while, for, if, switch, try/catch, return, break, throw comments
- use `//`, avoid `/**/`
- semicolons inserted if omitted (yikes!)
- declarations
- function scoping with `var`
- functions are expressions;
- closures (yippee!)

Variables

- A variable is a name associated with a piece of data
- Variables allow you to store and manipulate data in your programs
- Think of a variable as a mailbox which holds a specific piece of information

Variables

- In JavaScript variables are created using the keyword `var`
- Example:

```
var x = 10;
```

```
var y = 17;
```

```
var color = "red";
```

```
var name = "Katie";
```

Variables

- It is vitally important to distinguish between the *name* of the variable and the *value* of the variable
- For example, in the expression `var color="red"`, `color` is the name of the variable and `red` is the value. In other words, `color` is the name of the box while `red` is what is inside the box

Data Types

- Primitive Data Types
 - Numbers
 - Strings
 - Boolean (True, False)
- Composite Data Types
 - Arrays
 - Objects

Primitive Data Types

- **Numbers** - A number can be either an integer or a decimal
- **Strings** - A string is a sequence of letters or numbers enclosed in single or double quotes
- **Boolean** - True or False

Variables & Data Types

- JavaScript is *untyped*; It does not have explicit data types
- For instance, there is no way to specify that a particular variable represents an integer, string, or real number
- The same variable can have different data types in different contexts

Implicit Data Types

- Although JavaScript does not have explicit data types, it does have implicit data types
- If you have an expression which combines two numbers, it will evaluate to a number
- If you have an expression which combines a string and a number, it will evaluate to a string

Example: Variables

var x = 4;

Ans = x + y;

Ans => 15

var y = 11;

Ans = z + x;

var z = “cat”;

Ans => cat4

var q = “17”;

Ans = x + q;

Ans => 417

More Examples

```
var x = 4;
```

```
Ans = x + y + z;
```

```
Ans => 15cat
```

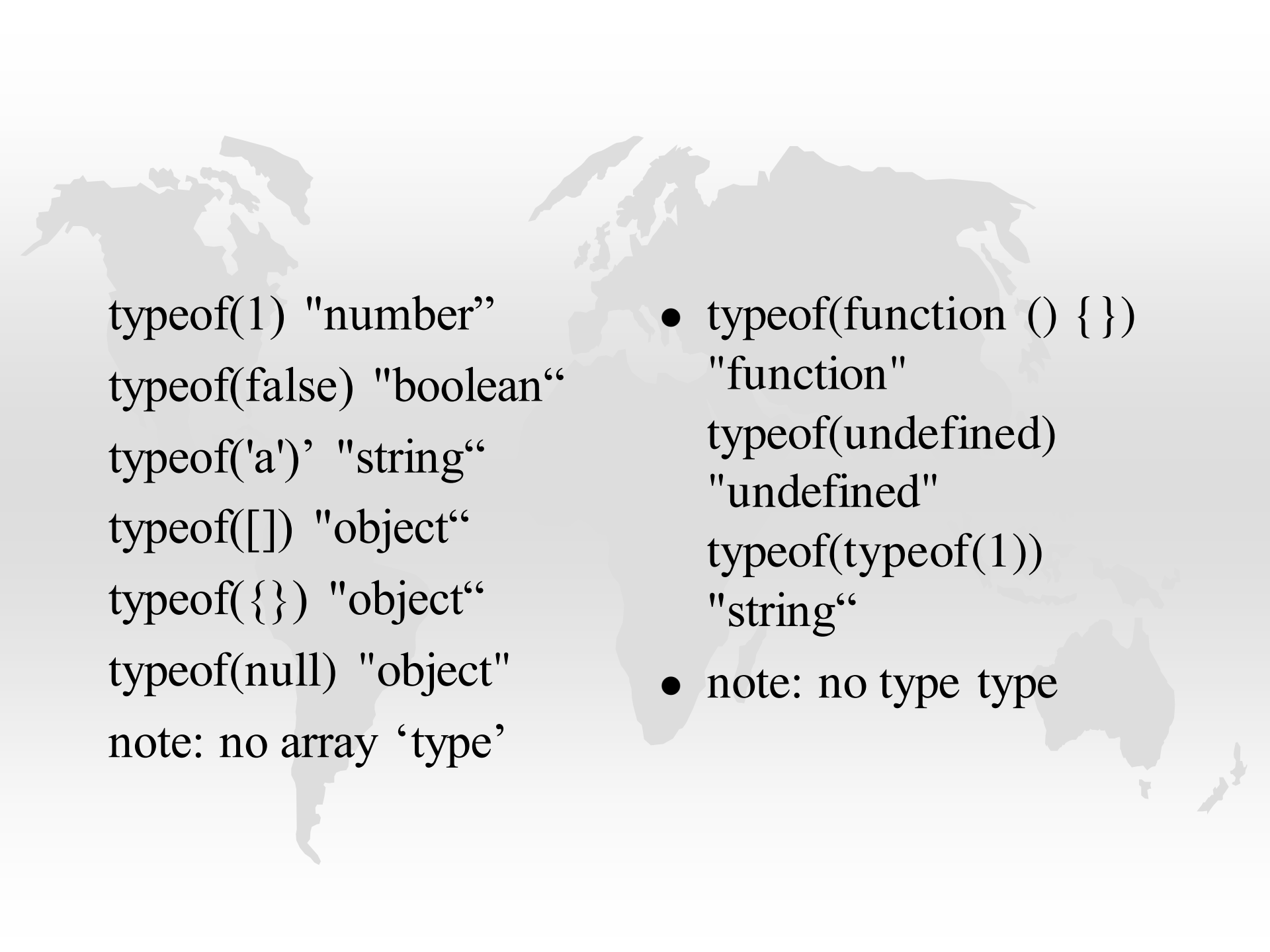
```
var y = 11;
```

```
Ans = q + x + y;
```

```
var z = "cat";
```

```
Ans => 17411
```

```
var q = "17";
```



typeof(1) "number"
typeof(false) "boolean"
typeof('a') "string"
typeof([]) "object"
typeof({}) "object"
typeof(null) "object"
note: no array 'type'

- typeof(function () {})
"function"
typeof(undefined)
"undefined"
typeof(typeof(1))
"string"
- note: no type type

- 
- funny values

undefined: lookup,
non-existent thing

null: special return
value

- equality

- use ===, !==

- Ex:

- 1 === 1 true

- 1.0 === 1 true

- 'hello' === 'hello' true

- [] === [] false

Arrays

- An array is a compound data type that stores numbered pieces of data
- Each numbered datum is called an *element* of the array and the number assigned to it is called an *index*.
- The elements of an array may be of any type. A single array can even store elements of different type.

Creating an Array

- There are several different ways to create an array in JavaScript
- Using the `Array()` constructor:
 - `var a = new Array(1, 2, 3, 4, 5);`
 - `var b = new Array(10);`
- Using array literals:
 - `var c = [1, 2, 3, 4, 5];`

Accessing Array Elements

- Array elements are accessed using the [] operator
- Example:
 - `var colors = ["red", "green", "blue"];`
 - `colors[0] => red`
 - `colors[1] => green`

Adding Elements

- To add a new element to an array, simply assign a value to it
- Example:

```
var a = new Array(10);
```

```
a[50] = 17;
```

Array Length

- All arrays created in JavaScript have a special length property that specifies how many elements the array contains
- Example:
 - `var colors = ["red", "green", "blue"];`
 - `colors.length => 3`

Primitive Data Types versus Composite Data Types

- Variables for primitive data types hold the actual value of the data
- Variables for composite types hold only references to the values of the composite type

A faint, light gray world map is visible in the background of the slide, centered behind the text.

Variable Names

- JavaScript is **case sensitive**
- Variable names cannot contain spaces, punctuation, or start with a digit
- Variable names cannot be reserved words

A faint, light gray world map is visible in the background of the slide, centered behind the text.

Programming Tips

- It is bad practice to change the implicit type of a variable. If a variable is initialized as a number, it should always be used as a number.
- Choose meaningful variable names

Statements

- A statement is a section of JavaScript that can be evaluated by a Web browser
- A script is simply a collection of statements

Examples:

```
Last_name = "Dunn";
```

```
x = 10 ;
```

```
y = x*x ;
```

Programming Tips

- It is a good idea to end each program statement with a semi-colon;
Although this is not necessary, it will prevent coding errors

- **Recommended:**

```
a = 3;  
b = 4;
```

- **Acceptable:**

```
a = 3; b = 4;
```

- **Wrong:**

```
a =  
3;
```

Operators

+ Addition
- Subtraction
* Multiplication
/ Division
% Modulus
++ Increment
-- Decrement

= = Equality
!= Inequality
! Logical NOT
&& Logical AND
|| Logical OR
? Conditional
Selection

Aggregate Assignments

- Aggregate assignments provide a shortcut by combining the assignment operator with some other operation
- The `+=` operator performs addition and assignment
- The expression `x = x + 7` is equivalent to the expression `x += 7`

Increment and Decrement

- Both the increment ($++$) and decrement ($--$) operator come in two forms: prefix and postfix
- These two forms yield different results

```
x = 10;      x = 10;  
y = ++ x;    z = x ++;
```

\Rightarrow **y = 11**

\Rightarrow **z = 10**

\Rightarrow **x = 11 in both cases**

Control Structures

- There are three basic types of control structures in JavaScript: the if statement, the while loop, and the for loop
- Each control structure manipulates a block of JavaScript expressions beginning with { and ending with }

The If Statement

- The if statement allows JavaScript programmers to make a decision
- Use an if statement whenever you come to a “fork” in the program

```
If ( x == 10)
{
    y = x*x;
}
else
{
    x = 0;
}
```

Repeat Loops

- A repeat loop is a group of statements that is repeated until a specified condition is met
- Repeat loops are very powerful programming tools; They allow for more efficient program design and are ideally suited for working with arrays

The While Loop

- The while loop is used to execute a block of code while a certain **condition** is true

```
count = 0;  
while (count <= 10) {  
    document.write(count);  
    count++;  
}
```

The For Loop

- The for loop is used when there is a need to have a **counter** of some kind
- The counter is initialized before the loop starts, tested after each iteration to see if it is below a target value, and finally updated at the end of the loop

Example: For Loop

// Print the numbers 1
through 10

```
for (i=1; i<= 10; i++)  
    document.write(i);
```

i=1 initializes the counter

i<=10 is the target
value

i++ updates the
counter at the end
of the loop

Example: For Loop

```
<SCRIPT  
  LANGUAGE=  
    "JavaScript">
```

```
document.write("1");  
document.write("2");  
document.write("3");  
document.write("4");  
document.write("5");  
</SCRIPT>
```

```
<SCRIPT  
  LANGUAGE=  
    "JavaScript">
```

```
  for (i=1; i<=5; i++)  
    document.write(i);
```

Functions

- Functions are a collection of JavaScript statement that performs a specified task
- Functions are used whenever it is necessary to repeat an operation

Functions

- Functions have inputs and outputs
- The inputs are passed into the function and are known as **arguments** or **parameters**
- Think of a function as a “black box” which performs an operation

Defining Functions

- The most common way to define a function is with the `function` statement.
- The function statement consists of the `function` keyword followed by the name of the function, a comma-separated list of parameter names in parentheses, and the statements which contain the body of the function enclosed in curly braces

Example: Function

```
function square(x)
{ return x*x; }
```

Name of Function: square

Input/Argument: x

```
z = 3;
```

```
sqr_z = square(z);
```

Output: $x*x$

Example: Function

```
function sum_of_squares(num1,num2)
{return (num1*num1) + (num2*num2);}
```

```
function sum_of_squares(num1,num2)
{return (square(num1) + square(num2));}
```