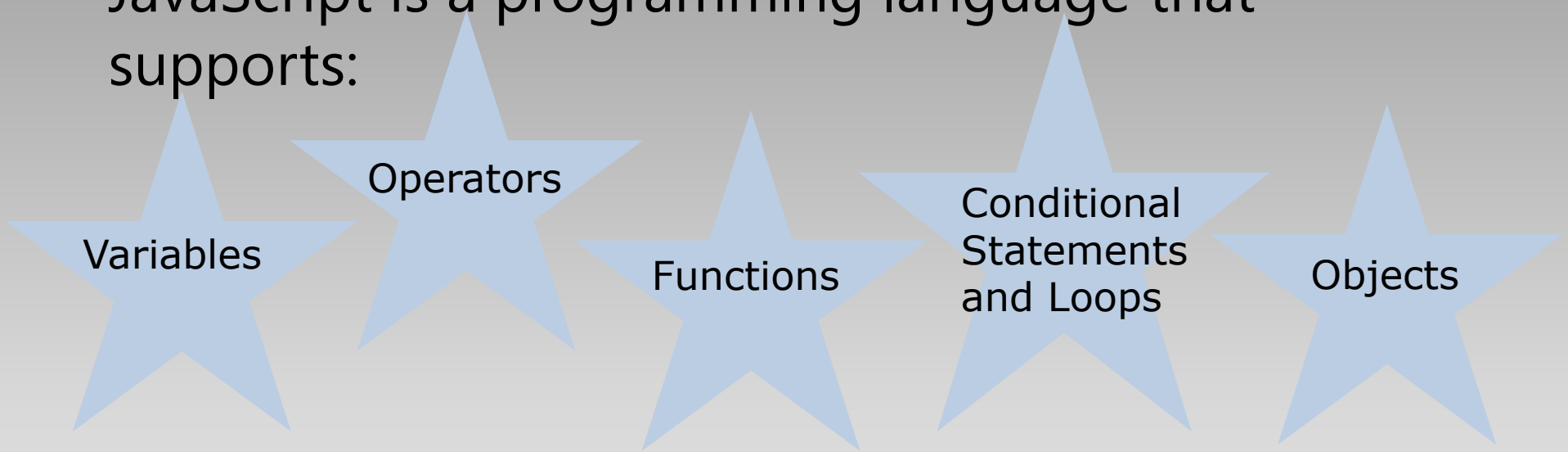
Introduction to JavaScript

Agenda

- JavaScript
- Document Object Model
- jQuery

What is JavaScript?

- JavaScript is a programming language that supports:



- Use JavaScript with the Document Object Model to make web pages dynamic.
- Use the AJAX API to make asynchronous requests to a web server.

JavaScript Syntax

- A JavaScript statement represents a line of code to be run
- Terminate statements with a semicolon

```
var thisVariable = 3;
counter = thisVariable + 1;
GoDoThisThing();
document.write("An incredibly really \
  very long greeting to the world");
```

- Use comments to add notes to your scripts

```
document.write("I'm learning JavaScript"); // display a message

/* You can use a multi-line comment
to add more information */
```

Variables, Data Types, and Operators

- Use **var** to declare variables

```
var answer = 3;
var actuallyAsString = "42";
```

- JavaScript has three simple types (Use only var)
  - String, Number, and Boolean

```
var noValue; // noValue has the value undefined
var nullValue = null; // null is different to undefined
```

- JavaScript supports many operators
  - Arithmetic, assignment, comparison and conditional

===

Always use === when comparing to any of these values.

```
var zero = 0;
var emptyString = "";
var falseVar = false;
zero == falseVar; // returns true converts to same type before comparing;
zero === falseVar; // returns false;
emptyString == falseVar;  // returns true;
emptyString === falseVar; // returns false;
```

Functions

- Functions are named blocks of reusable code:

```
function aName( argument1, argument2, …, argumentN )
{
  statement1;
  statement2;
  …
  statementN;
}
```

  - Arguments are only accessible inside the function
  - A function can return a value
  - A function can also declare local variables
  - Global variables defined outside of a function are available to all functions in scripts referenced by a page

Conditional Statements

- JavaScript provides two conditional constructs
  - if:

```
if (TotalAmountPaid > AdvancePaid) {
   GenerateNewInvoice();
} else {
  WishGuestAPleasantJourney();
}
```

  - switch:

```
var RoomRate;
switch (typeOfRoom) {
  case "Suite":
    RoomRate = 500;
    break;
  case "King":
    RoomRate = 400;
    break;
  default:
    RoomRate = 300;
}
```

Looping Statements

- JavaScript provides three loop constructs
  - while:

```
while (GuestIsStillCheckedIn())
{
  numberOfNightsStay += 1;
}
```

  - do while:

```
do {
  eatARoundOfToast();
} while (StillHungry())
```

  - for:

```
for (var i=0; i<10; i++) {
  plumpUpAPillow();
}
```

  - forin:

```
for (var i in Array) {
        var item = Array[i];
}
```

Using Object Types

- JavaScript has a number of built-in object types:
  - String, Date, Array, RegExp

```
var seasonsArray = new Array("Spring", "Summer", "Autumn", "Winter");
var seasonsArray = ["Spring", "Summer", "Autumn", "Winter"];
…
var autumnLocation = seasonsArray.indexOf("Autumn");
```

```
var re = new RegExp("[dh]og");
if (re.test("dog")) {…}
```

- JavaScript also provides singleton types providing useful functionality:
  - Math - Math.Pi, Math.random

Defining Arrays of Objects by Using JSON

- JSON is a format for serializing objects:

```
var attendees = [
    {
        "name": “Eric Gruber",
        "currentTrack": "1"
    },
    {
        "name": “Martin Weber",
        "currentTrack": “2"
    }
]
```

- JavaScript provides APIs for serializing and parsing JSON data

Document Object Model

- The DOM provides a programmatic API for controlling a browser and accessing the contents of a web page:
  - Finding and setting the values of elements on a page
  - Handling events for controls on a page
  - Modifying the styles associated with elements
  - Validating and updating web pages

Finding Elements in the DOM

- Given the following form:

```
<form name="contactForm">
  <input type="text" name="nameBox"   id="nameBoxId" />
</form>
```

  - You can reference the form by using:

```
document.forms[0]   // forms is a zero-based array
document.forms["contactForm"]
document.forms.contactForm
document.contactForm
```

  - You can reference the **nameBox** text box by using:

```
document.forms.contactForm.elements[0]
document.forms.contactForm.elements["nameBox"]
document.forms.contactForm.nameBox
document.contactForm.nameBox
document.getElementById("nameBoxId")
```

Adding, Removing, and Manipulating Objects in the DOM

To modify an element on a page:

1. Create a new object containing the new data.
2. Find the parent element that should contain the new data.
3. Append, insert, or replace the data in the element with the new data.

To remove an element or attribute:

1. Find the parent element.
2. Use **removeChild** or **removeAttribute** to remove the data.

Handling Events in the DOM

- The DOM defines events that can be triggered by the browser or by the user
- Many HTML elements define callbacks that run when an event occurs:

```
var helpIcon = document.getElementById("helpIcon");

helpIcon.addEventListener("mouseover",
        function() { window.alert('Some help text'); }, false);
```

```
document.images.helpIcon.onmouseover =
  function() { window.alert('Some help text'); };
```

- To remove an event listener:

```
helpIcon.removeEventListener("mouseover", ShowHelpText, false);
```

The jQuery Library

- jQuery provides portability for JavaScript code, enabling you to easily build cross-browser web applications:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>jQuery Example</title>
    <script type="text/javascript" src="Scripts/jquery-1.8.0.min.js">
    </script>
</head>
<body>
    …
    <script type="text/javascript">
        $(document).ready(function () {
            // some code
        });
    </script>
</body>
</html>
```

Selecting Elements and Traversing the DOM by Using jQuery

- jQuery uses the same selector syntax as CSS

```
<script type="text/javascript">
    $(document).ready(function () {
        $("h2").each(function () {
            this.style.color = “red";
        });
    });
</script>
```

- jQuery provides additional functions for traversing and filtering elements

Adding, Removing, and Modifying Elements by Using jQuery

- Use the **selector** function to specify the elements to change or remove

- Common methods include:
  - addClass
  - append
  - detach
  - html
  - replaceWith
  - val

```
$("p").addClass("strike");
$("ul").append("<li>New item</li>");
$(“#Warning").detach();
$("div").html("<span>Hello</span>");
$(“#Warning").replaceWith("<p>Panic over!</p>");
$("input[type=text").val();
```

Handling Control Events by Using jQuery

- Use the jQuery **selector** function to find the item that raises the event
- Use the **bind** method (or a jQuery shortcut) to bind the event handler to the event

```
<script type="text/javascript">
    $(document).ready(function () {
        $("#submit").click(
            function () {
                var userName = $("#NameBox").val();
                $("#thankYouArea").replaceWith(
                        "<p>Thank you " + userName + "</p>");
        })
    });
</script>
```