JavaScript

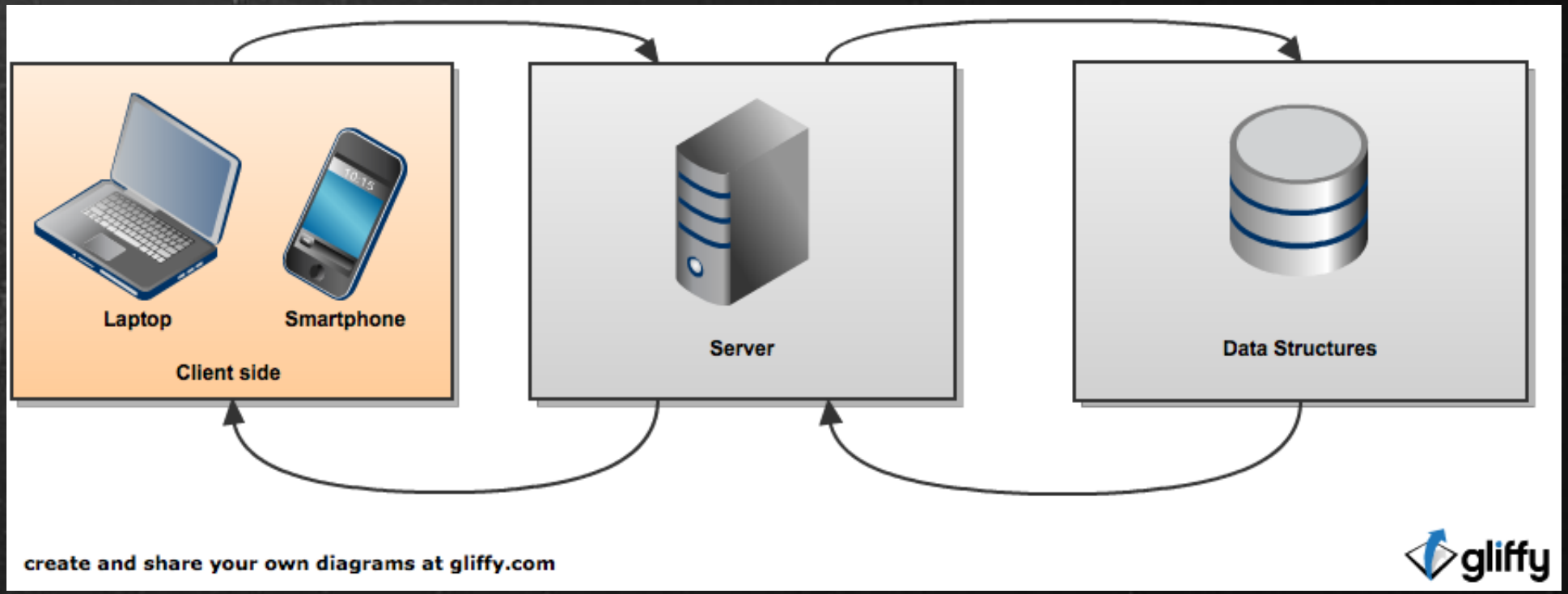
Wasim Ahmad Khan

Outline

- JavaScript
  - Introduction
  - Syntax details: variables, types, conditionals, loops
  - Using the console
  - Accessing HTML
  - Modifying HTML
  - Debugging JavaScript

What is JavaScript

- JavaScript is text that is fed into a browser that can interpret it and then it is enacted by the browser
- Because JavaScript code can run locally in a user's browser (rather than on a remote server), the browser can respond to user actions quickly, making an application more responsive
- Works with document structure created by HTML


Laptop
Smartphone
Client side
Server
Data Structures
create and share your own diagrams at gliffy.com
gliffy

JavaScript

- Drag and Drop
- Form validation
- Dynamic drop-down menus (http://sandbox.scriptiny.com/dropdown-menu/)
- Pop-up windows
- Alerts

Java is to JavaScript
what
Car is to Carpet

Using JavaScript

1. Inline
2. In the <head> tag of your HTML
3. Within the <body> tag of HTML
4. By including an external file (.js)

Using JavaScript

1. Inline

```
<html>
 <head>
  ...
 </head>
 <body>
  ...
  <a href="javascript: alert()"></a>
  <a href="#" onclick="alert()"></a>
  ...
 </body>
</html>
```

Using JavaScript

2. In the <head> tag of your HTML

```
<html>
 <head>
  ...
  <script type="text/javascript">
   alert();
  </script>
  ...
 </head>
 <body>
  ...
 </body>
</html>
```

Using JavaScript

3. Within the <body> tag of HTML

```
<html>
 <head>
  ...
 </head>
 <body>
  ...
  <script type="text/javascript">
   alert();
  </script>
 </body>
</html>
```

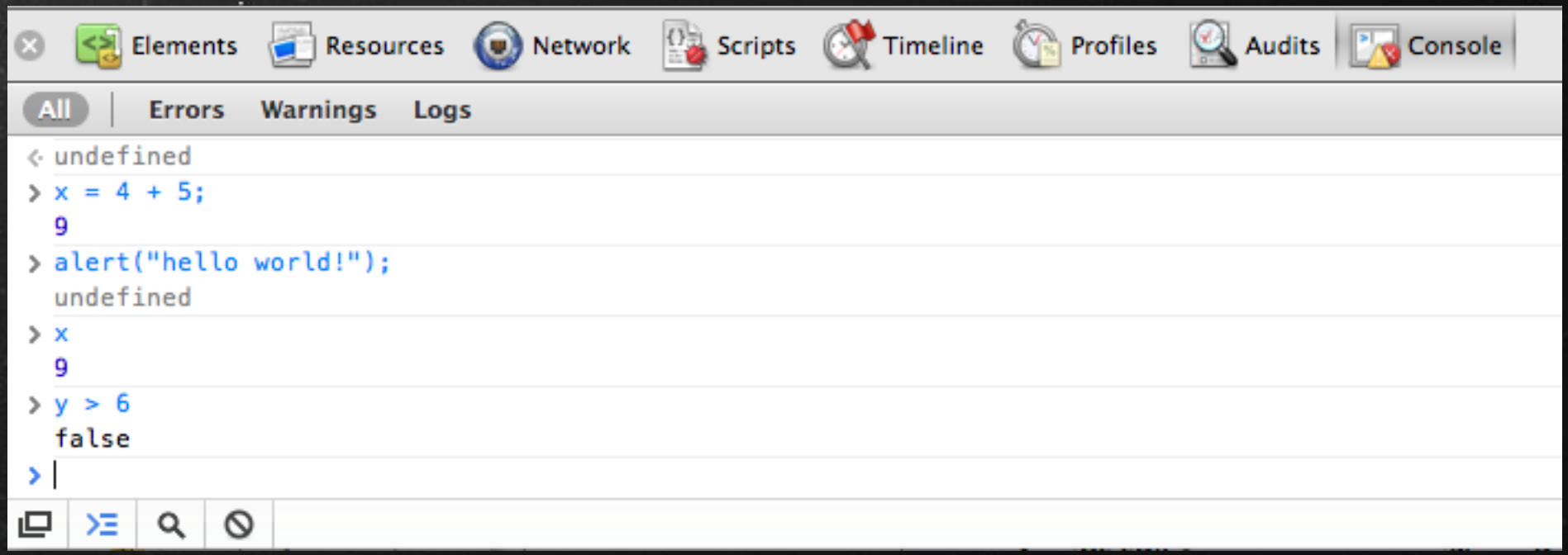
Using JavaScript

4. By including an external file

```
<html>
 <head>
  ...
 </head>
 <body>
  ...
  <script type="text/javascript"
       src="my_javascript_file.js">
  </script>
 </body>
</html
```

JavaScript in Console

1. Open Developer Tools in Chrome (right-click > Inspect Element)
2. Test your JavaScript in the browser!



JavaScript Syntax

- The <script> tag tells the browser to run JavaScript for the text included between the tags
- Each JavaScript line is an instruction sent to the browser.
- Each JavaScript line ends in a semicolon (;)
- You can group several lines of code together with curly braces ({ .... })

JavaScript Variables

```
var car = "ford";
var x = 0;
var y;
```

- Variable can have a value associated with it
- The value of the variable can change during the execution
- The keyword "var" has to be used in order to declare the variable.
- Declaring the variable is done at the top of the script

JavaScript Operations

```
x = 5 + 34;
y = 3 - 2;
result = 200 / 40;
speed = (43 * (23 + 43)) / 4;
x++; // x = x + 1;
y--; // y = y - 1;
x+=y; // x = x + y;
```

JavaScript Comments

```
// This is a comment


/*
    This is a ...
        two-line comment!
*/
```

JavaScript Variable Types

- Numbers
  - var x = 0;
- Strings - used for representing text
  - var message = "Please try again."
  - var car = 'Ford'
  - You can use either "" or ''
- Arrays - a list of values
  - var arr = [3, 4, 5];
  - var pets = new Array();

JavaScript Strings

- Used for variables that hold text
- You can add strings together

```
var text = "I love " + "mushrooms";

var vegetable = "kale";

text = "I love " + vegetable;
    // The value of text will be "I love kale"
```

JavaScript Arrays

```
var pets=new Array();
pets[0]="cat";
pets[1]="dog";
pets[2]="fish";

var pets=new Array("cat","dog","fish");

var pets=["cat","dog","fish"];
```

JavaScript Arrays

```
var fave_pet = pets[2]  // will get "fish"

pets[2] = parrots; // This reassigns the value

var pet_count = pets.length; // Returns 3
```

JavaScript Conditionals

```
if (condition) {
  // code will run if condition is true
}
```

JavaScript Conditionals

```
if (condition){
  // code will run if condition is true
}
else {
  // code will run if condition is false
}
```

JavaScript Conditionals

```
if (condition1) {
  // code will run if condition1 is true
}
else if (condition2) {
  // code will run if condition2 is true
}
else {
 // code will run if neither condition1 nor condition2 is true
}
```

JavaScript Conditionals

- `x == 4` // is value of x equal to number 4?
- `x != 4` // is value of x different from number 4?
- `x > 4` // is value of x greater than number 4?
- `x < 4` // is value of x smaller than 4?
- `x >= 4` // is value of x greater or equal to number 4?
- `x <= 4` // is value of x smaller or equal to 4?

```
if (x == 4) {
   ......
}
```

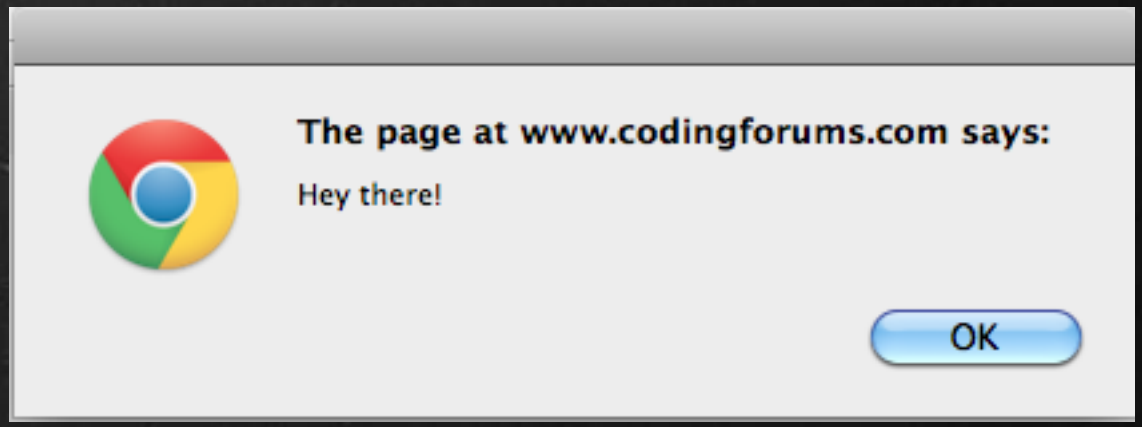
JavaScript For Loops

```
for (var i=0; i<5; i++) {
   // lines of code to run for each element
  // i = 0 first time loop runs
  // i = 1 second time, i = 2 the third time
   .....
}
```

- Declare a variable named "i"
- Assign the value of 0 to variable i
- Set the upper limit for i. In this case, i cannot exceed 5.
- i++ increments i by 1 each time the loop runs

JavaScript Alert

```
alert ("Hey there!");
```



JavaScript Functions

- A function contains a block of code that will only run when the function is called

Declaring a function:

```
function someFunction (variable1, variable2,..) {
  // Code in here will run
}
```

JavaScript Functions

To call a function:

```
someFunction (value1, value2, ...);
    // declared by us

alert ("Hello world!");
    // function native to JavaScript
```

JavaScript Functions

```
<html>
<head>
<script type="text/javascript">
function displayMessage() {
  alert("Confirmed!");
}
</script>
</head>
<body>
<a href="#" onclick="displayMessage()">Click me</a>
</body>
</html>
```

JavaScript Events

- JavaScript detects when a certain action is taking place.
- Examples of actions detected:
  - The mouse was clicked
  - The mouse moves
  - The mouse is over an HTML element
  - User pressed a key
  - A page has loaded
  - A page is resizing

JavaScript Events

Mouse is over an HTML element

- <a href="#" onmouseover="alert('You are over link!')">...

User clicks on the HTML element

- <a href="#" onclick="alert('You are clicking link!')">...

JavaScript Events

The body of the HTML has loaded:

```
<html>
<head>
<script type="text/javascript">
function bodyLoaded() {
   alert("Page is loaded");
}
</script>
</head>
<body onload="bodyLoaded()">
   ....
</body>
</html>
```

JavaScript and HTML

```
<p>Once upon a time...</p>
<p>What a great story!</p>

document.getElementsByTagName("p");
```

JavaScript and HTML

```
<div id="title">Once upon a time...</div>
```

```
document.getElementById("title");
```

JavaScript and HTML

```
<a href="#" class="utility">Link</a>
<a href="#" class="utility">Another Link</a>
```

```
document.getElementsByClassName("utility");
```

JavaScript and HTML

Other properties:

- parentNode
- childNodes
- firstChild
- lastChild
- nextSibing
- previousSibling

JavaScript and HTML

```
<div id="story">
  <p id="title">Once upon a time...</p>
</div>

 var p = document.getElementById("title");

 var parent = p.parentNode;    // will be <div id="story">
```

JavaScript and HTML

```
<p>Once upon a time...</p>
<p>What a great story!</p>

paragraphs = document.getElementsByTagName("p");
                    //Array

paragraphs.length; // Returns 2 paragraphs in array

for (i=0; i<paragraphs.length; i++) {
    ...
}
```

JavaScript and HTML

- document.anchors // Array containing all links
- document.anchors[2] // Third link element on page (<a>)
- document.forms[0] // Returns the first form HTML element
- document.forms // Array of all form elements on page
- document.images.length // How many image tags on page
- document.images // Array of all image elements on page

JavaScript and HTML

Get the contents of an HTML element:

```
<html>
<head>
<script type="text/javascript">
function getValue() {
  var header_value = document.getElementById("header");
  alert(header_value.innerHTML);
}
</script>
</head>
<body>
  <h1 href="#" id="header" onclick="getValue()">Click me!</h1>
</body>
</html>
```

JavaScript and HTML

Return the class name of an HTML element:

```
<div id="title" class="large">Savings</div>

document.getElementById("title").className;
   // Will return "large"
```

JavaScript and HTML

Create HTML elements dynamically:

```
var body = document.getElementsByTagName('body')[0];
var newdiv = document.createElement('div');

newdiv.setAttribute('id',"images");

newdiv.innerHTML =
   "Images: <img src='plant.jpg' onclick='alert('Plant!')'/>";

body.appendChild(newdiv);
```

JavaScript & HTML & CSS

Change the CSS style of an HTML element:

```
<div id="title" class="large">Savings</div>

document.getElementById("title").style.fontSize = "14px;"
```

- element.style.color
- element.style.background
- element.style.backgroundImage
- element.style.backgroundPosition

JavaScript & HTML & CSS

Making an HTML element appear and disappear:

```
<div id="title" class="large">Savings</div>

document.getElementById("title").style.display = "none";
document.getElementById("title").style.display = "block";
```

JavaScript & HTML & CSS

CSS selectors are separated by dash - :

- background-image
- font-size

In Javascript, they are camelCased:

- backgroundImage
- fontSize

Debugging JavaScript

```
var image = "cat.jpg";

console.log("Value of var 'image' is ", image);
console.log("Value of var 'image' is " +image);
```

This prints in Developer Inspector tools Console tab:

DOM

- History / Idea
  - HTML and XML documents consist of tags
  - Well-formatted documents (required in XHTML and XML) can be viewed as a tree
    - Ex: http://www.w3schools.com/htmldom/default.asp
  - DOM provides a language-independent, object-based model for accessing / modifying and adding to these tags
  - DOM 0
    - Not formally specified by W3C but includes a lot of useful functionality

DOM

- DOM 1, 2, 3
  - Formal specifications of model and functionality
  - Each builds on / improves previous
- Unfortunately there are still compatibility issues with browsers
  - IE through IE8 does not fully support DOM 2
    - It has its own syntax for event attachment
  - IE9 does support DOM 2 but until all older versions go away the problem still exists

Events

- With documents DOM specifies events and event handlers
  - Event model is similar to the one used in Java
  - Different parts of a document have different events associated with them
  - We can define handlers to react to these events
- These allow us to "interact" with and add "dynamic content" to web documents
  - Ex: Can preprocess form elements
  - Ex: Can load / update / change what is displayed in response to an event

DOM and Events

- document refers to the top-level document
  - Each document has access to its properties and to the components that are declared within it
    - Ex: title, URL, forms[], applets[], images[]
    - Attributes with IDs can also be specified by ID (from DOM 1)
  - Once we know the components, events and event-handlers, we can write JavaScript programs to process Web pages on the client-side
    - Client computers are typically less busy than servers, so whatever we can do at the client will be helpful overall

DOM and Events

  - Ex: Checking form correctness before it is submitted
- In HTML documents events are specified through tag attributes
  - Within the tag identifying an HTML component, we can specify in an attribute how the component reacts to various events
- See Sebesta Table 5.1 for events and tag attributes and Table 5.2 for the tags that have a given attribute
- Similar in idea to Java, we assign event handling code to the tag attributes, and the code executes when the event occurs

Events

- We can also attach events in Javascript
  - In DOM 0, events are attached in an "inline" way:
    - Ex: theElement.onclick = functionName
  - In DOM 2, a more flexible event model was developed, so that more than one handler could be attached to the same event:
    - Ex: theElement.addEventListener(type, fn, opt)
      - Where opt is a boolean to determine if the event is "captured" or "bubbles"
        - » See: http://en.wikipedia.org/wiki/DOM_Events
    - Unfortunately, IE (up through IE8) does not use DOM 2
      - It has its own, similar model

DOM and Events

- Ex: Event mouseover occurs when the mouse is place over a displayed element
  - Elements that allow for the mouseover event have the attribute onmouseover
  - In HTML or Javascript, the programmer assigns a function call to the attribute, so that when the event occurs the function is called

```
<input type = "radio" name = "choice" value = "1"
  onmouseover = "showChoice(1)">
```

  - We can define showChoice however we'd like
    - ex: alert("You are about to choose Choice 1");

Example: Pre-processing a Form

- A very common client-side operation is pre-processing a form
  - Ensure that fields are filled and formatted correctly, so server does not have to
    - Saves load on the server, saves time and saves bandwidth
    - We can check a form overall by using the attribute onsubmit
      - We can put it right into the form as an attribute
      - Or we can assign the attribute through the document object in Javascript

Example: Pre-processing a form

- We can check individual components as they are entered as well
  - Ex: <input type = "text"> has the onchange attribute
    - » Triggered when contents are changed and focus changes
  - Ex: <input type = "radio"> has the onclick attribute
    - » Triggered when the radio button is clicked with the mouse

JavaScript Exercises

1. Make a few boxes using CSS and HTML. When you click on them, change their color.
2. Choose a random different color every time, use an array to store them.

Resources

- Code Academy JavaScript
  Tutorial: http://www.codecademy.com/
- W3Schools JavaScript
  examples: http://www.w3schools.com/js/js_ex_dom.asp