JavaScript’s role on the Web

- JavaScript Programming Language
  - Developed by Netscape for use in Navigator Web Browsers
  - Purpose → make web pages (documents) more dynamic and interactive
    - Change contents of document, provide forms and controls, animation, control web browser window, etc.
Introduction

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language i.e. lightweight programming language
- A JavaScript is usually embedded directly in HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
  - Converts code to executable format each time it runs
  - Converted when browser loads a web document
- JavaScript is supported by all major browsers
- Java and Javascript are not the same although they share some common syntax
What can a JavaScript Do?

- Gives HTML designers a **programming tool**
  - HTML authors are normally not programmers, but JavaScript is a scripting language with a **very simple syntax**!
  - Almost anyone can put small "snippets" of code into their HTML pages
- Puts **dynamic text** into an HTML page
  - A JavaScript statement like this: 
    ```javascript
document.write("<h1>" + name + "</h1>"
    ```
  - can write a variable text into an HTML page
- React to **events**
  - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- Can **read and write HTML elements**
- Can be used to **validate data** (e.g. form data) before it is submitted to a server
Elements of JavaScript

- Typical programming elements:
  - variables, operators, expressions, statements, definition of functions

- Connection to Web
  - built-in functions (e.g., Date())
  - Document Object Model = names of “things”, properties, methods (e.g., document.write)
  - events (e.g., onSubmit, onMouseOver)
JavaScript and HTML pages

- Inserting JavaScript into an HTML page

```html
<html>
<body>
  <script type="text/javascript">
    document.write("Hello World!")
  </script>
</body>
</html>
```

- Note that semicolons are optional.
JavaScript and HTML pages

- JavaScript is usually placed in the HEAD section
  - Make sure that the function is loaded before used
    ```html
    <html>
    <head>
        <script type="text/javascript">
            some statements
        </script>
    </head>
    <body>
        Some HTML
    </body>
    </html>
    ```
- You can place any number of scripts in one page
- Can write code to a file and load the file
  ```html
  <html>
  <head>
      <script src="xxx.js"></script>
  </head>
  <body> </body>
  </html>
  ```
JavaScript Variables

- Used to store information
- Variable name are case sensitive
- Must begin with a letter or the underscore
  - Example:
    ```javascript
    var strname = "Chris"
    
    Or
    
    strname = "Chris"
    ```
- The scope of variables is as usual
  - Local within the functions
  - Global (page level) if declared outside the functions
# JavaScript Operators

## Arithmetic Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
<td>x=2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x+2</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>x=2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5-x</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>x=4</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x*5</td>
<td></td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>15/5</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5/2</td>
<td>2.5</td>
</tr>
<tr>
<td>%</td>
<td>Modulus (division remainder)</td>
<td>5%2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10%8</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10%2</td>
<td>0</td>
</tr>
<tr>
<td>++</td>
<td>Increment</td>
<td>x=5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>x++</td>
<td>x=6</td>
</tr>
<tr>
<td>--</td>
<td>Decrement</td>
<td>x=5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>x--</td>
<td>x=4</td>
</tr>
</tbody>
</table>

## Assignment Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Is The Same As</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>x=y</td>
<td>x=y</td>
</tr>
<tr>
<td>+=</td>
<td>x+=y</td>
<td>x=x+y</td>
</tr>
<tr>
<td>-=</td>
<td>x-=y</td>
<td>x=x-y</td>
</tr>
<tr>
<td>*=</td>
<td>x*=y</td>
<td>x=x*y</td>
</tr>
<tr>
<td>/=</td>
<td>x/=y</td>
<td>x=x/y</td>
</tr>
<tr>
<td>%=</td>
<td>x%=?</td>
<td>x=x%?y</td>
</tr>
</tbody>
</table>
# JavaScript Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>==</code></td>
<td>is equal to</td>
<td><code>5==8</code> returns false</td>
<td><code>&amp;&amp;</code></td>
<td>and</td>
<td><code>x=6, y=3</code> ((x &lt; 10 &amp;&amp; y &gt; 1)) returns true</td>
</tr>
<tr>
<td><code>!=</code></td>
<td>is not equal</td>
<td><code>5!=8</code> returns true</td>
<td>`</td>
<td></td>
<td>`</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>is greater than</td>
<td><code>5&gt;8</code> returns false</td>
<td><code>!</code></td>
<td>not</td>
<td><code>x=6, y=3</code> !(x==y) returns true</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>is less than</td>
<td><code>5&lt;8</code> returns true</td>
<td><code>&gt;=</code></td>
<td>is greater than or equal to</td>
<td><code>5&gt;=8</code> returns false</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>is less than or equal to</td>
<td><code>5&lt;=8</code> returns true</td>
<td><code>&lt;=</code></td>
<td>is less than or equal to</td>
<td><code>5&lt;=8</code> returns true</td>
</tr>
</tbody>
</table>
Writing formatted text

```html
<html>
<body>
<script type="text/javascript">
document.write("<h1>Hello World!</h1>"
</script>
</body>
</html>
```
As in Java
Example:

```javascript
let txt1 = "What a very";
let txt2 = "nice day!";
let txt3 = txt1 + " " + txt2;
```

txt3 contains “What a very nice day!”

Creating functions

```javascript
function myfunction(argument1, argument2, etc) {
    some statements
}
```

Functions with returning values

```javascript
function result(a, b) {
    let c = a + b;
    return c;
}
```

Calling Functions

```javascript
myfunction(argument1, argument2, etc);
```
```html
<script type="text/javascript">
// If the time on your browser
// is less than 12, you will get
// a "Good morning" greeting.

var d=new Date()
var time=d.getHours()
if (time<10) {
    document.write("<b>Good morning</b>")
} else {
    document.write("<b>Good afternoon</b>")
}
</script>
```

**If and If..Else statement**

**Loops**

- while
  ```javascript
  while (condition) {
      code to be executed
  }
  ```

- do...while
  ```javascript
  do {
      code to be executed
  } while (condition)
  ```

- for
  ```javascript
  for (initial; condition; increment) {
      code to be executed
  }
  ```
Switch

```javascript
<script type="text/javascript">
// You will receive a different greeting based on what day it is. Note that Sunday=0, Monday=1, Tuesday=2, etc.

var d=new Date()
theDay=d.getDay()
switch (theDay) {
    case 5:
        document.write("Finally Friday")
        break
    case 6:
        document.write("Super Saturday")
        break
    case 0:
        document.write("Sleepy Sunday")
        break
    default:
        document.write("I'm looking forward to this weekend!")
}
</script>
```
JavaScript Events

This lets the hyperlink take place.

This is an event.

Note single quotes inside double quotes
var input = prompt("What is your name?", "Chris");
print("Well hello " + (input || "dear"));

- If the user presses 'Cancel' or closes the prompt dialog in some other way without giving a name, the variable input will hold the value **null or ""
- Both of these would give **false** when converted to a boolean
- The expression **input || "dear"** can in this case be read as ‘the value of the variable input, or else the string “dear”’
  - It is an easy way to provide a ‘fallback’ value.
**Error Handling**

- Similar to Java → `try {…} catch (error) {…} finally {…}` statements

  ```javascript
  function lastElement(array) {
    if (array.length > 0)
      return array[array.length - 1]
    else
      throw "Can not take the last element of an empty array."
  }

  function lastElementPlusTen(array) {
    return lastElement(array) + 10
  }
  ...
  try {
    print(lastElementPlusTen([]));
  } catch (error) {
    print("Something went wrong: ", error)
  } finally {
    print("Finished")
  }
  ```
### Manipulating HTML Page Elements

- **Document Object Model (DOM)**
  - Contains all HTML elements loaded in a web page
  - Can be used by scripting languages to manipulate HTML on the fly
  - DOM objects:

<table>
<thead>
<tr>
<th>Anchor (hyperlink)</th>
<th>Select</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body</td>
<td>Style</td>
</tr>
<tr>
<td>Button</td>
<td>Submit</td>
</tr>
<tr>
<td>Checkbox</td>
<td>Table</td>
</tr>
<tr>
<td>Document</td>
<td>TableData</td>
</tr>
<tr>
<td>Form</td>
<td>TableHeader</td>
</tr>
<tr>
<td>Frame</td>
<td>TableRow</td>
</tr>
<tr>
<td>Frameset</td>
<td>Text</td>
</tr>
<tr>
<td>Image</td>
<td>Textarea</td>
</tr>
<tr>
<td>Link</td>
<td>Textarea</td>
</tr>
<tr>
<td>Location</td>
<td>Window</td>
</tr>
<tr>
<td>Password</td>
<td></td>
</tr>
<tr>
<td>Radio</td>
<td></td>
</tr>
</tbody>
</table>
Element Manipulation: Anchor (Changing URL and text)

```html
<html>
<head>
    <script type="text/javascript">
        function myHref() {
            document.getElementById('myAnchor').innerText = "Visit W3Schools"
            document.getElementById('myAnchor').href = "http://www.w3schools.com"
        }
    </script>
</head>
<body>
    <a id="myAnchor" href="http://www.microsoft.com">Visit Microsoft</a>
    <form>
        <input type="button" onclick="myHref()" value="Change URL and text"/>
    </form>
</body>
</html>
```
Element Manipulation: Forms

```html
<html>
<head>
<script type="text/javascript">
function showFormElements(theForm) {
  str="Form Elements: 
  for (i=0; i<theForm.length; i++)
    str+="
        +theForm.elements[i].name
  alert(str)
}
</script>
</head>
<body>
<form>
  First name: <input type="text" name="fname" size="20">
  Last name: <input type="text" name="lname" size="20">
  <br />
  <input type="button" name="button1" value="Display name of form elements"
         onClick="showFormElements(this.form)">
</form>
</body>
</html>
```
Element Manipulation: Submitting Forms

```html
<html>
<head>
<script type="text/javascript">
function formSubmit() {
    document.forms.myForm.submit()
}
</script>
</head>
<body>
<form name="myForm" action="sample2.asp" method="get">
    Firstname: <input type="text" name="firstname" size="20"><br/>
    Lastname: <input type="text" name="lastname" size="20"><br/> <br/>
    <input type="button" onclick="formSubmit()" value="Submit">
</form>
</body>
</html>
```
Element Manipulation: Validating Forms

```html
<html>
<head>
<script type="text/javascript">
function validate() {
  x=document.myForm
  at=x.email.value.indexOf("@")
  if (at == -1) {
    alert("Not a valid e-mail")
    return false
  }
}
</script>
</head>
<body>
<form name="myForm"
  action="tryjs_submitpage.htm"
  onsubmit="return validate()">
  Enter your E-mail:
  <input type="text" name="email" size="20">
  <input type="submit" value="Submit">
</form>
<p><b>Note:</b> This example ONLY tests if the e-mail address contains an "@" character. A "real-life" code would have to test for punctuations, spaces and other things as well.</p>
</body>
</html>
```
Javascript Resources

- http://www.javascript.com
- http://www.pageresource.com/jscript/
- http://javascript.internet.com/
- http://javascriptkit.com/
- http://www.w3schools.com/js/default.asp

- Use Google
Create a page with JavaScript to do the following. These can all be on one page.

- Prompt the user for their name.
- Use a pop-up box to welcome the user by name.
- Display the current date on the page in the following format: October 30, 2000. Do not display the time. Do not "hard code" the date; if I load the page tomorrow, I should get a different date than if I load it today.
- Display the last modified date of the document.
- Put some useful advice, on any subject, in the status line of the browser.
- Display a message saying Good Morning if it is in the morning, Good Afternoon if it is in the afternoon, and Good Evening if it is in the Evening.
Practice 2

- **JavaScript Form Validation Exercise**
  - Create a form that includes 2 textboxes (one for name, one for email address), a textarea, a dropdown box (<select> tag), and a set of checkboxes. Format the form using a table. The ACTION attribute of the form tag should be to email the form results to your email address. Feel free to use a preexisting form rather than creating one for this exercise.
  - Add JavaScript event handlers to the form to validate the form before it is emailed to you. If there is a problem, display the problem with the page in some way. Possible ways to do this are:
    • a pop-up box
    • displaying the problems in a new window in red text and having the user use the back button to go back to the original page
    • displaying the problems in the current window at the top of the page in red text
  - Validate that an entry has been made in each form element
  - Validate that the email address contains an @ sign.
- If you have time and want to try something more, add a second drop down list and make this second one dynamic, i.e. make the choices that appear in the second drop down list depend on what choice is made in the first list.