

Lesson #13:

JavaScript II

Date and Time

Forms

Strings

Date and time



- The Date object is useful when you want to display a date or use a timestamp in some sort of calculation. You can either make a Date object by supplying the date of your choice, or you can let JavaScript create a Date object based on your visitor's system clock (not the server's).
- It is important to note that if someone's clock is off by a few hours or they are in a different time zone, then the Date object will create a different time from the one created on your own computer.

Date and time



```
var currentTime = new Date()
```

- The Date object has been created, and now we have a variable that holds the current date. To get the information we need to print out, we have to utilize some or all of the following functions:
- `getTime()` - Number of milliseconds since 1/1/1970 @ 12:00 AM
- `getSeconds()` - Number of seconds (0-59)
- `getMinutes()` - Number of minutes (0-59)
- `getHours()` - Number of hours (0-23)
- `getDay()` - Day of the week(0-6). 0 = Sunday, ... , 6 = Saturday

Date and time



- getDate() - Day of the month (0-31)
- getMonth() - Number of month (0-11)
- getFullYear() - The four digit year (1970-9999)

```
<script type="text/javascript">
```

```
var currentTime = new Date()
```

```
var month = currentTime.getMonth() + 1
```

```
var day = currentTime.getDate()
```

```
var year = currentTime.getFullYear()
```

```
document.write(month + "/" + day + "/" + year)
```

```
</script>
```

11/5/2008

Form validation

- There's nothing more troublesome than receiving orders, guest book entries, or other form submitted data that are incomplete in some way. You can avoid these headaches once and for all with JavaScript's form validation.
- The idea behind JavaScript form validation is to provide a method to check the user entered information before they can even submit it. JavaScript also lets you display helpful alerts to inform the user what information they have entered incorrectly and how they can fix it.



Form Validation: Checking for Non-Empty

- This has to be the most common type of form validation. You want to be sure that your visitors enter data into the HTML fields you have "required" for a valid submission.

```
<script type='text/javascript'>
```

```
function notEmpty(elem, helperMsg){
```

```
    if(elem.value.length == 0){
```

```
        alert(helperMsg);
```

```
        elem.focus();
```

```
        return false; }
```

```
    return true;}
```

```
</script>
```

function
parameters



Form Validation: Checking for Non-Empty

```
<form>
```

```
Enter your name: <input type="text" id="req1" />
```

```
<input type="button"
```

```
  onclick="notEmpty(document.getElementById('req1'  
  '), 'You must enter your name!)" value="Check
```

```
Field" />
```

```
</form>
```

- As long as `elem.value.length` isn't 0 then it's not empty and we return true, otherwise we send an alert to the user with a `helperMsg` to inform them of their error and return false.



Form Validation: Checking for all numbers

```
function isNumeric(elem, helperMsg){  
    var numericExpression = /^[0-9]+$/;  
    if(elem.value.match(numericExpression)){  
        return true;  
    }else{  
        alert(helperMsg);  
        elem.focus();  
        return false;  
    }  
}
```

See course website for
regular expressions
in JavaScript!



Form Validation: Length restriction

- Being able to restrict the number of characters a user can enter into a field is one of the best ways to prevent bad data. For example, if you know that the postal code field should only be 7 characters you know that 2 characters is not sufficient.

```
<script type="text/javascript">  
function lengthRestriction(elem, min, max){  
    var ulInput = elem.value;  
    if(ulInput.length >= min && ulInput.length <= max){  
        return true;  
    }else{  
        alert("Please enter between " +min+ " and " +max+ "  
characters");  
        elem.focus();  
        return false;}}  
</script>
```



Form Validation: Length restriction

- Being able to restrict the number of characters a user can enter into a field is one of the best ways to prevent bad data. For example, if you know that the postal code field should only be 7 characters you know that 2 characters is not sufficient.

```
<form>
```

```
Postal Code? <input type="text" id="pc" />
```

```
<input type="button"
```

```
  onclick="lengthRestriction(document.getElementById('pc'), 6,  
  7)"
```

```
  value="Validate Postal Code" />
```

```
</form>
```



Form Validation: Length restriction

- Every email is made up for 5 parts: A combination of letters, numbers, periods, hyphens, plus signs, and/or underscores, the @ symbol, a combination of letters, numbers, hyphens, and/or periods, a period and the top level domain (com, net, org, ca, in, ...).

```
function emailValidator(elem, helperMsg){  
    var emailExp = /^[\\w\\-\\.\\+]+\\@[a-zA-Z0-9\\.\\-]+\\.?[a-zA-z0-9]  
    {2,4}$/;  
    if(elem.value.match(emailExp)){  
        return true;  
    }else{  
        alert(helperMsg);  
        elem.focus();  
        return false;}}
```



Form Validation: FINAL VALIDATION

- The final step is to be able to perform all of these validation steps when the user is ready to submit their data.
- Each form has a JavaScript event called onSubmit that is triggered when its submit button is clicked. If this event returns 0 or false then a form cannot be submitted, and if it returns 1 or true it will always be submitted.
- `<form onSubmit="return formValidator()" >`
- A bunch of nested ifs validating each form element and calling other functions (including the ones we saw already) will constitute the formValidator function.
- Try to write one to validate a phone number (all digits), a postal code and make sure no field is empty



Strings: length property

- length: the length property returns the number of characters that are in a string, using an integer

```
<script type="text/javascript">
```

```
var myString = "the quick brown fox";
```

```
document.write("The string is this long: " +  
myString.length);
```

```
myString = myString + " jumped over the lazy dog.";
```

```
document.write("<br />The string is now this long: " +  
myString.length);
```

```
</script>
```

```
The string is this long: 19  
The string is now this long: 45
```



Strings: split function

- The ability to split up a string into separate chunks is available in JavaScript as well. If you have a long string like "Toronto Ottawa London Hamilton Windsor" and want to store each city name separately, you can specify the space character " " and have the split function create a new chunk (an array cell) every time it sees a space.

```
<script type="text/javascript">
```

```
var cities = "Toronto Ottawa London Hamilton Windsor";
```

```
var cityArray = cities.split(" ");
```

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

```
    document.write("<br /> City #" + i + " = " + cityArray[i]); }
```

```
</script>
```

```
City #0 = Toronto  
City #1 = Ottawa  
City #2 = London  
City #3 = Hamilton  
City #4 = Windsor
```



Strings: search function

- This string function takes a regular expression and then examines that string to see if there are any matches for that expression. If there is a match, it will return the position in the string where the match was found. If there isn't a match, it will return -1.

```
<script type="text/javascript">
```

```
var cities = "Toronto Ottawa London Hamilton Windsor";
```

```
var regexp = /on|to/;
```

```
var found = cities.search(regexp);
```

```
if(found != -1)
```

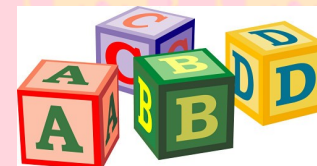
```
    document.write("Found at position " + found);
```

```
else
```

```
    document.write("No match");
```

```
</script>
```

Found at position 3



Strings: replace function

- The string replace function has two arguments: what word is going to be replaced (this can be a string or a regular expression) and what the word will be replaced with (this needs to be a string).
- *replace* returns the new string with the replaces, but if there weren't any words to replace, then the original string is returned.

```
<script type="text/javascript">
```

```
var cities = "Toronto Ottawa London Hamilton Windsor";
```

```
var modcities = cities.replace(/on/g, "?");
```

```
document.write ("Cities: " + cities);
```

```
document.write ("<br />Modified cities: " + modcities);
```

```
</script>
```

```
Cities: Toronto Ottawa London Hamilton Windsor  
Modified cities: Tor??to Ottawa L??d?? Hamilt?? Windsor
```



Changing HTML

- Each HTML element has an *innerHTML* property that defines both the HTML code and the text that occurs between that element's opening and closing tag. By changing an element's *innerHTML* after some user interaction, you can make much more interactive pages.
- However, using *innerHTML* requires some preparation if you want to be able to use it easily and reliably. First, you must give the element you wish to change an id. With that id in place you will be able to use the *getElementById* function, which works on all browsers.
- After you have that set up you can now manipulate the text of an element. To start off, let's try changing the text inside a bold tag.

Changing HTML

```
<script type="text/javascript">
```

```
function changeText(){
```

```
    document.getElementById('visitor').innerHTML =  
    "Homer Simpson";
```

```
}
```

```
</script>
```

```
<p>Welcome to the site <span id="visitor">Peter  
Griffin</span>! </p>
```

```
<input type="button" onclick="changeText()"  
value="Change Name" />
```

Changing HTML

```
<script type="text/javascript">
```

```
function changeText(){
```

```
    var name = document.getElementById('name').value;
```

```
    document.getElementById('visitor').innerHTML = name;
```

```
}
```

```
</script>
```

```
<p>Welcome to the site <span id="visitor">Peter  
Griffin</span>! </p>
```

```
<input type="text" id="name" value="Enter Name Here" />
```

```
<input type="button" onclick="changeText()  
value="Change Name" />
```

Welcome to the site Peter Griffin!

Changing HTML

```
<script type="text/javascript">
```

```
function changeCol(){
```

```
    var oldHTML =
```

```
    document.getElementById('para').innerHTML;
```

```
    var newHTML = "<span style='color:#cc0000'>" +  
    oldHTML + "</span>";
```

```
    document.getElementById('para').innerHTML =  
    newHTML;
```

```
}
```

```
</script>
```

```
<p id='para'>Welcome to the site my friend!</p>
```

```
<input type="button" onclick="changeCol()"  
    value="Change Text" />
```



End of lesson