CS 3710 Vocational Languages

October 17, 2012

Some Window Commands

- `window.alert("Hello World!!");`
- `window.prompt("What is your name?");`
- `window/document.write("Hello " + window.prompt("What is your name?"); "enter your name"));

```javascript
var x = window.prompt("Enter your first number: ", "enter here");
var y = window.prompt("Enter your second number: ", "enter here");
var z = x*1.0+y*1.0;
window/document.write("The sum of "+ x + " and "+ y + " is ");
window/document.write(z);
```

Using Buttons for JavaScripts

```
<FORM><INPUT TYPE="button" VALUE="Blue"
onClick="window/document.bgColor='blue'"></FORM>
- `<FORM>`: creates a form to use a button.
- `<INPUT>`: create an input area of some kind.
- `type="button"`: declares the input area to be a button.
- `value="Blue"`: declares the text on the button.
- `onClick="..."`: gives the action once the button is clicked, which can be a function call.
```

Fundamental Datatypes vs. Objects

- **Fundamental Datatypes:**
  - Numbers, strings, boolean, null and undefined
  - They have methods.
  - They are literals and are immutable.
- **Objects:** **mutable**
  - Arrays
  - Functions
  - Regular expressions
  - Objects
Fundamental data type

- Numbers have methods:
  - Math object has a set of methods that act on numbers.
    Math.floor(-5.2) is ?.
- Each string has a length property:
  - "seven".length is ?.
  - Strings have methods:
    "tax".toUpperCase() === ?
- The following give Boolean false value:
  - false, null, undefined, empty string '', number 0
  - if("") 3; else 5; //?

Objects

- JavaScript Objects are class-free.
- An object is a container of properties.
- A property has a name and a value.
- Object properties can be another object, which can be function, array or object.
- We can create objects hierarchy using "prototype linkage", hence allows one object inherits the properties of another object.

An Object Example

```javascript
var flight = {
  airline: "Oceanic",
  number: 815,
  departure: {
    IATA: "SYD",
    time: "2004-09-22 14:55",
    city: "Sydney"
  },
  arrival: {
    IATA: "LAX",
    time: "2004-09-23 10:42",
    city: "Los Angeles"
  },
  travel: function(){return ...}
};
```

Retrieve Object Property Values

- Use dot () or [] to retrieve the property values of an object:
  - flight.airline //?
  - flight[departure] //?
  - flight.departure.time //?
  - flight.travel //?
- Retrieve the value of a non-existent property gives value undefined.
  - flight.status //?
```javascript
var flight = {
  airline: "Oceanic",
  number: 815,
  departure: {
    IATA: "SYD",
    time: "2004-09-22 14:55",
    city: "Sydney"
  },
  arrival: {
    IATA: "LAX",
    time: "2004-09-23 10:42",
    city: "Los Angeles"
  },
  travel: function(){return ...}
};
```
### Update Object Property Values

- For existing property name, the new assigned value overwrite the old value.
  ```javascript
  var flight = {
    airline: 'Oceanic',
    number: 815,
    departure: {
      IATA: 'SYD',
      time: '2004-09-22 14:55',
      city: 'Sydney'
    },
    arrival: {
      IATA: 'LAX',
      time: '2004-09-23 10:42',
      city: 'Los Angeles'
    },
    travel: function(){return ...
  };
  ```

- For non-existing property name, the new property name and value are added to the object.
  ```javascript
  flight.equipment = {model: 'Boeing 777'};
  flight.status = 'delayed';
  ```

### JavaScript manipulates object references for computation (like Java)

```javascript
var newflight=flight;
newflight.status = 'arrived';
```

### Object Prototype

- Every object is linked to a `prototype` from which it can inherit properties.
  ```javascript
  var newflight = Object.create(flight);
  newflight.status = 'arrived';
  ```

### Enumerate Object Properties

- `for-in` enumerates all property names, including functions and prototype properties, of an object.
- We are mostly interested in data, not function of an object. To filter out the unwanted values:
  ```javascript
  for (aName in flight){
    if (typeof flight[aName] !== 'function'){
      document.writeln(aName);
    } //output?
  }
  ```
Delete Object Properties

- The `delete` operator removes a property from an object if the property exist.
- It does not touch any of the objects in the prototype linkage.

```javascript
delete flight.number;
flight.number; //undefined
```