Knowledge Base for Javascript Programming Language

- Book: ***Learning Javascript*** 2rd ED (Shelley Powers)
- Sources:

- Book notes:

  - Not all script embedded in web pages is JavasScript
    - text/ecmascript
    - text/jscript
    - text/vbscript
    - text/vbs
  - `function`

    ```
    function functionname(params){}
    ```

  - Event Handlers
    - onclick
    - onmouseover
    - onmouseout
    - onfocus
    - onblur
  - An identifier must begin with a character, dollar sign, or underscore (Unicode letter since 1.5)
  - Case-sensitive, can't be a JS key word
  - Browser Object Model (BOM), such as *document* and *window*, *reserved words in browers*: **alert eval location open array focus math outerHeight blur name parent history navigator parseFloat date image number regExp document isNaN object status escape length onLoad string**
  - CamelCase notation: lowercase for the fist letter and the capatial letter for the following words
  - Underscore is used to signal a variable that's an objet's private data member `var _break = someval;`
  - Primitive Types: just three: `string`, `numeric`, `boolean`
  - Built-in objects: **String**, **Number**, and **Boolean**. The String object wraps the string primitive--just as the Number and Boolean objects wrap their individual primitive types--when using the primitive type like an object.
    - `var strString = "This is a string";` `var another = 'This is \"also\" a string';` `"\u7231"`
    - **+** The number is convertted to a string and then the two string are concatenated
    - If the string is the first in a sequence of values, all of the number that follow are treated as

strings `var strValue = "4" + 3 + 1; // become "431"`, `var strValueTwo = 4+ 3 + "1"; //become 71`

- If use operators other than +, the opposite type of conversion is applied -- string is convertted to a number `var first = ""35" - 3"; // 32`, `var second = 30 / "3"; // 10,` `var third = "3" * 3; //9`

- *toString conversion table* Input | Reslut --- | --- Undefined |"undefined" Null | "null" Boolean | if true, then "true",; if false, then "false" Number | Thre string representation of the number, or NaN if the variable hods this latter values String | No conversion

- Example

```
function convertToString(){
 var newNumber = 34.56;
 var newBoolean = true; // false
 var nothing;
 var newNull = null;

 var strNumber = String(newNumber); var strBoolean = String(newBoolean);
 var strUnderfined = String(nothing); var strNull  = String(newNull);
 document.writeln(strOutput);
}
```

The Boolean Data Type

- `true` and `false`
- `Boolean(somevar);`

The Number Data Type

- Two special numbers exxist: positive and negative infinity by `Infinity` tand `-Infinity`
- Range: `-2e31` to `2e31`

`null` and `undefined`

- if you have declared but not initialized the variable, it is consided undefined: `var undefString;`
- `if (sValue) ...` if not null and initialized (undefined), true; otherwise false
- `if (isNan(sValue)) ...` if string cannot be implicitly converted to number, return true

Constants: Named But Not Variables

- `const CURRENT_MONTH = 3.5 ;` treated it as a read-only value from that time forward

- Javascript constant has global and local scope
- Name is in uppercase

Operators and Statements Resume at page 59, Chapter 3

- Semicolon is not required

```
condition ? value if true: value if false;
```

- And `&&` , Or `||` , `if (!n(nValue > 10))`
- `while` and `for`

```
while (nValue <=10 ) {
 strValue+= nValue;
 bValue++;
}

do{
   startValue += nValue;
   nValue++;
} while (nValue <=10)

for (initial valuel condition; update){}
for (var i =0 ; i< 10; i++){
 document.writeln("hello");
}

\\ loop over object property or array
var MyText = {
one: "one",
two : "two",
three: "three"
};

for (var prop in MyText){
document.writeln(prop + "<br />");
}

var tsts = new Array('one', 'two','three'); // for array, just loop over
for (indx in tsts){
   alert(tsts[indx]);
}
```

Objects

- Objects are those that parallel out data types, `String` for strings, `Boolean` for booleans, `Number` for numbers

```
var myName = "Shelley"; // Define a variable primitive
alert(myName.length);  // after the initialization, when the String method
//can access its property
alert(myName.strike()); // access its method
var myName2 = new String("Shelley"); // explicitly create a String object
alert(myName2.valueOf()); // or directly call its name to return the value
var myName3 = String("Shelley"); // it is equvialent to create a primitive
```

- If you create a string primitive and then access it like an object, JavaScript will convert the primitive to an object when you access a `String` property, but it does so by converting the primitive to a temporary `String` object, and then discarding the object when it's finished with the property. This isn't efficient.
- `Boolean` object, inherit *`toString`* and *`valueOf`* methods from higher object

```
var boolflag = new Boolean(); // value is false
var boolflag = new Boolean(1); //value is true
var boolflag = new Boolean(false);
var boolflag = new Boolean(""); // set to false for empty String
var boolfag = new Boolean("false"); // set to true
```

some text