Built-ins

User Interfaces

420-CW4-AB

Built-in Functions

Built-in functions are functions that are already built into JavaScript.

They extend the flexibility of the language.

There are popular common function as well as four built-in for objects:
Array, Date, Math, Number, and String

Each built-in function has special-purpose, it also has properties and methods associated with it.

Array

```
arrayVar.length;    // number of elements in array
arrayVar.isArray(); // returns true if it's an array
arrayVar.concat( X ); // returns combined arrays
arrayVar.indexOf( X ); // returns first index of X
arrayVar.join( X );
   // combines elements into string using X as delimeter
arrayVar.slice(X, Y);
   // extracts Y elements from array starting at X
```

Array – Mutator Methods

```
arrayVar.pop();     // Remove last element
arrayVar.push();    // Add element at end
arrayVar.reverse(); // Reverses order of array
arrayVar.shift();   // Remove first element
arrayVar.sort();    // Sorts the array
arrayVar.splice();  // Add/Remove elements
arrayVar.unshift(); // Add element at start
```

Built-in Functions

isNaN()

- Evaluates a variable and checks verifies if it is NOT a number.

```
isNaN( expression );
```

- Returns true, if the variable is Not a Number.
  Returns false, if the variable is a number.
  Careful it's easy to get this one confused

```
isNaN("Hello");   // true
isNaN(23);        // false
isNaN("12.45");   // false
isNaN("");        // false
isNaN(true);      // false
isNaN(false);     // false
isNaN(null);      // false
isNaN(undefined); // true
```

Built-in Functions

parseFloat()
parseInt()

- Parses a string argument and returns a float value

```
parseFloat( argument );
```

- Parses a string argument and returns an integer value

```
parseInt( argument );
```

- If the variable begins with a number, the function reads through the variable until it finds the end of a valid the number; it then cuts off the rest and returns the result.
- If the parameter does not begin with a number, the function returns NaN.

```
parseFloat("Hello"); // NaN
parseFloat("123.33");    // 123.33
parseInt(1.2333);    // 1
parseFloat("3.4Five"); // 3.4
parseInt("one23");   // NaN
parseFloat([]);  // NaN
```

Number

```
myVar.toFixed( X );
// Returns a string with X digits
    after the decimal

let numFixed = 1.2345;

numFixed.toFixed();      // 1
numFixed.toFixed(5)      // 1.23450
numFixed.toFixed(2);     // 1.23
numFixed.toFixed(1);     // 1.2
```

```
myVar.toPrecision( X );
// Returns a string with X digits
    in the entire number

let  numPrec  = 1.2345;

numPrec.toPrecision();   // 1.2345
numPrec.toPrecision(5);  // 1.2345
numPrec.toPrecision(2);  // 1.2
numPrec.toPrecision(1);  // 1
```

String

```
myStr.length;           // number of character in string
myStr.charAt( X );      // returns character at X index
myStr.indexOf( X );     // return first index of X
myStr.lastIndexOf( X ); // return last index of X
myStr.substring( X[, Y]); // returns part of the string
myStr.toLowerCase();  // returns string in lowercase
myStr.toUpperCase();   // returns string in uppercase
```

Math

```
Math.E;  // Euler's constant (2.718...)
Math.PI; // constant PI (3.14159...)

Math.abs( X );     // returns absolute value of X
Math.ceil( X );    // returns largest integer value >= X
Math.floor( X );   // returns smallest integer value <= X
Math.random();     // returns random number between 0 and 1
Math.round( X );   // returns number rounded to nearest integer
Math.trunc( X );   // returns int value (removes after decimal)
```

Random Number between 0 and *n*

```
/**
 * Generate a random int between 0 and max (exclusive)
 * @param {*} max
 */
function getRandomIntFrom0(max) {
  return Math.floor(Math.random() *  max);
}
```

Random Number between 1 and *n*

```
/**
 * Generate a random int between 1 and max (inclusive)
 * @param {*} max
 */
function getRandomIntFrom1(max) {
  return Math.floor(Math.random() *  max) + 1;
}
```

Generate Random Number

```
/**
 * Generate random int between min and max(both inclusive)
 * @param {*} max
 * @param {*} [min=1]
 */
function getRandomInt(max, min=1) {
  return Math.floor(Math.random() *  (max - min + 1)) + min;
}
```

Date Methods

```
Date.now();    // returns current Unix timestamp

Date.parse( X );

   // returns a string that is an attempt to
      convert X into Unix timestamp
```

- A Unix timestamp is a number that represents the time, in milliseconds, that have elapsed since Jan 1 1970 0 hours, 0 minutes and 0 seconds
- 1524493800 is equivalent to April 23rd 2018 at 2:30pm (UTC)

Date

- The Date objects can only be initialized by calling JavaScript Date as a constructor.
- calling it as a regular function (i.e. without **new**) will return a string, not the Date object;

```
let myDate = new Date();
let myDate = new Date(value);  //value is a unix timestamp
let myDate = new Date(dateString);  //string representation
let
myDate = new Date(year, monthIndex [, day [, hours [, minutes
 [, seconds [, milliseconds]]]]]);
   // monthIndex is 0-based. This means Jan = 0 and Dec = 11.
   // if day is 0 it will be set to the last day of the month
```

Date - Getters

```
let d = new Date();
d.getDate();        // Returns the day of the month (1-31)
d.getDay();         // Returns the day of the week (0-6) Sunday=0
d.getFullYear();    // Returns the year (0-2018+)
d.getMonth();       // Returns the month (0-11)

d.getHours();       // Returns the hour (0-23)
d.getMinutes();     // Returns the minutes (0-59)
d.getSeconds();     // Returns the seconds (0-59)

d.getTime();        // Returns the Unix Timestamp
```

Date - Setters

```
let d = new Date();
d.setDate();        // Set the day of the month (1-31)
d.setFullYear();    // Set the year (0-2018+)
d.setMonth();       // Set the month (0-11)

d.setHours();       // Set the hour (0-23)
d.setMinutes();     // Set the minutes (0-59)
d.setSeconds();     // Set the seconds (0-59)

d.setTime();        // Set the Unix Timestamp
```

Questions ?

The important thing is not to stop questioning. Curiosity has its own reason for existence.
– Albert Einstein