Desarrollo de Aplicaciones Web Empresariales

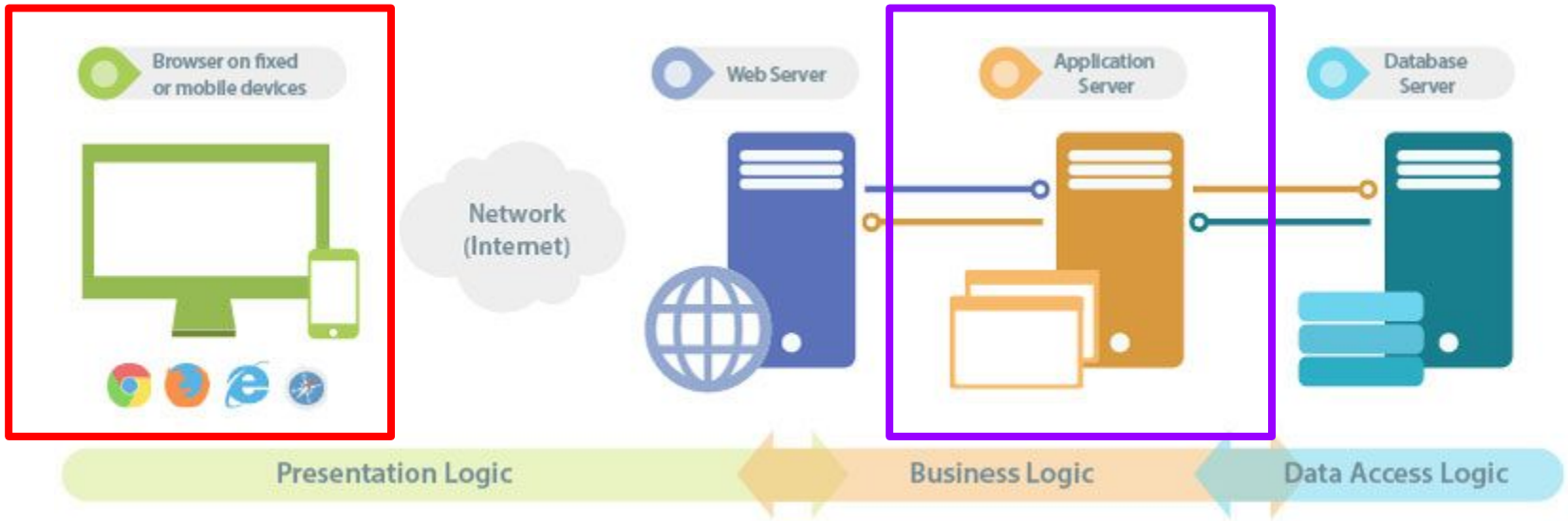
Exequiel Fuentes Lettura
exequiel.fuentes@ucn.cl

Información de contacto

- Exequiel Fuentes Lettura
  - Email: exequiel.fuentes@ucn.cl
  - Horario de Atención: Jueves y Viernes, bloque C

- Departamento de Ingeniería de Sistemas y Computación
  - Oficina: Y1 - 329
  - http://www.disc.ucn.cl

Arquitectura de aplicaciones Web: Javascript, lo básico



Material: https://developer.mozilla.org/en-US/docs/Web/JavaScript

Qué es Javascript

- Un LP de alto nivel, dinámico, interpretado, débilmente tipado.
- Soporta: orientación a objetos, programación imperativa, programación funcional.
- Tiene una API para trabajar con textos, arreglos, fechas y expresiones regulares.
- No es similar a Java, es más cercano a C.
- También conocido como ECMAScript.

Tipeo dinámico

```
var i;
typeof i == 'undefined'
i = 32; // typeof i == typeof 32 == 'number'
i = "foobar"; // typeof i == typeof 'foobar' == 'string'
i = true; // typeof i == 'boolean'
```

- La variables adquieren el tipo de su última asignación.
- Tipos de primitivas: undefined, number, string, boolean, function, object

Alcance de las variables

Dos alcances: Global y local a la función

```
var globalVar;

function() {
    var localVar;
    if (globalVar > 0) {
        var localVar2 = 2;
    }
}
```

Alcance "*hoisted*"

```
function foo() {
    var x;
    x = 2;
}

// Lo mismo que arriba:
function foo() {
    x = 2
    var x;
}
```

Problemas con el alcance

- No es buena idea usar variables globales, sobre todo en browsers. Es fácil tener conflictos entre módulos.
- *Hoisting* puede causar confusión en alcances locales
  - function() {
    for(var i = 0; i < 10; i++) {
    ...
    for(var i = 0; i < 25; i++) { // Error: i already defined
- Se sugiere definir todas las variables al principio de la función.
- ES6 introduce **let** para definir alcances específicos.

Tipo: number

- El tipo number es almacenado como punto flotante (double en C).
  - MAX_INT = (2^{53} - 1) = 9007199254740991
- Algunas cosas extrañas: NaN, Infinity son number.
  - 1/0 == Infinity
  - Math.sqrt(-1) == NaN
- Otra cosa:
  - (0.1 + 0.2) == 0.3 es falso // 0.30000000000000004
  - operadores bitwise (ejemplo: ~, &, |, ^, >>, <<, >>>) son 32 bit!

Tipo: String

- Variable de tipo texto
  - var s = 'Esto es un ejemplo'; // se puede usar “Esto es un ejemplo”
- El signo + puede concatenar strings.
- Hay muchos métodos útiles: indexOf(), charAt(), match(), search(), replace(), toUpperCase(), toLowerCase(), slice(), substr(), ….

Tipo: boolean

- Puede ser: true o false
- Valores falsos son:
  - false, 0, "", null, undefined, and NaN
- Valores verdaderos son:
  - No falso! (todos los objetos, strings no vacíos, numbers distintos de cero, functions, etc.)

Tipo: undefined y null

- **undefined**: Significa que no tiene valor definido o declarada.
- **null:** Utilizado por el programador para algo específico. Es un valor, de tipo object.
- Ambos son falsos pero no iguales:
  - null === undefined; // false
  - null !== undefined // true

Tipo: function

```
var fac = function fac(x) {
    if (x <= 1) return 1;
    return x*fac(x-1);
}
typeof fac == 'function' // true
fac.name == 'fac' // true
```

- Permite pasar argumentos.
- Argumentos no especificados, tienen valor undefined.
- Todas las funciones retornan un valor, si no hay un return explícito retorna undefined.

Otras formas

```
function myFunc(routine) {
    console.log('Called with', routine.toString());
    var retVal = routine(10);
    console.log('retVal', retVal);
}

myFunc(function (x) {
    console.log('Called with', x);
    return x+1;
});
```

Tipo: object

- Un object es una colección no ordenada de pares clave-valor llamada propiedades:
  - var p1 = {};
    var p2 = {nombre: "Exequiel", edad: 37, ciudad: "Antofagasta"};
- La clave puede ser un string (incluso vacío):
  - var x = { "": "empty", "---": "dashes"}
- Para obtener el valor, se utiliza la clave:
  - p2.nombre o p2["nombre"]
    x["---"]
    p1.nonExistent == undefined

Operaciones sobre object

- Para agregar:
  - var persona = {};
    persona.nombre = "Javier";
- Para remover:
  - var persona = {nombre: "Javier"};
    delete persona.nombre; // persona es ahora un object vacío
- Para enumerar:
  - Object.keys({nombre: "Ignacio", edad: 25}) // ["nombre", "edad"]

Arreglos

- Es un tipo especial de object
  - var anArr = [1,2,3];
    typeof anArr == 'object'
- Los índices son enteros no negativos.
- Se puede almacenar cualquier elemento:
  - anArr[5] = 'hola'; // [1,2,3,,,'hola']
- Tienen varios métodos útiles: length, push, pop, shift, unshift, sort, reverse, splice, ….
- Pero tienen cosas extrañas:
  - var anArr = [1,2,3];
    anArr.name = 'Exequiel'; // [1, 2, 3, name: 'Exequiel']
  - Cuidado con length! => anArr.length = 0 // [name: 'Exequiel']

Tipo: Date

- Es un tipo especial de object
  - var date = new Date();
    typeof date == 'object'
- Es un número en milisegundos desde la medianoche de 1 de Enero de 1970 UTC.
- Es necesario definir el Timezone.
- Hay varios métodos para retornar y fijar el objeto:
  - date.valueOf() = 1452359316314
  - date.toISOString() = '2016-01-09T17:08:36.314Z'
  - date.toLocaleString() = '1/9/2016, 9:08:36 AM'

Expresiones regulares

- Define un patrón para ser buscado en un string:
  - var re = /ab+c/;
    var re2 = new RegExp("ab+c");
- Para string se puede utilizar los métodos: search(), match(), replace() y split()
- Útil para:
  - Buscar.
  - Parsear.

Expresiones regulares - exec/match/replace

```
var str = "This has 'quoted' words like 'this'";
var re = /'[^']*'/g;

re.exec(str);    // Returns ["'quoted'", index: 9, input: …
re.exec(str);    // Returns ["'this'", index: 29, input: …
re.exec(str);    // Returns null

str.match(/'[^']*'/g);    // Returns ["'quoted'", "'this'"]
str.replace(/'[^']*'/g, 'XXX');      // Returns: 'This has XXX words with XXX.'
```

Expresiones regulares - search/test

```
/HALT/.test(str);     // Returns true if string str has the substr HALT
/halt/i.test(str);     // Same but ignore case
/[Hh]alt [A-Z]/.test(str);     // Returns true if str either “Halt L” or “halt L”

'XXX abbbbbbc'.search(/ab+c/);     // Returns 4 (position of ‘a’)
'XXX ac'.search(/ab+c/);     // Returns -1, no match
'XXX ac'.search(/ab*c/);     // Returns 4
'12e34'.search(/[^\d]/);     // Returns 2, esto es una negación
'foo: bar;'.search(/...\s*:\s*...\s*;/);     // Returns 0
```

- https://developer.mozilla.org/en/docs/Web/JavaScript/Guide/Regular_Expressions

Excepciones - try/catch

- Los errores se reportan a través de excepciones.
- Termina la ejecución del programa con un error.
- Se atrapa la excepción con try/catch:

```
try {
    nonExistentFunction();
} catch (err) { // typeof err 'object'
    console.log("Error llamando a la funcion ", err.name, err.message);
}
```

Excepciones - throw/finally

- Se emite una excepción usando throw

```
try {
    throw "Help!";
} catch (errstr) { // errstr === "Help!"
    console.log('Got exception', errstr);
} finally {
    // Este bloque siempre se ejecuta despues de try/catch
}
```

- Una convención utilizada es:
  - console.log("Got Error:", err.stack || err.message || err);

Cargar el script en la página

- Siempre debe estar definido al final del documento.

- Incluyendo un archivo separado:
  - <script type="text/javascript" src="script.js"></script>

- En la misma página:

```
<script type="text/javascript">
//<![CDATA[
Escriba su código Javascript aca
//]]>
</script>
```

Tarea: Agregar y quitar 1

- En este ejercicio debes desarrollar una función que permita agregar o quitar 1 a la cantidad que está en el elemento input. La función se llama modificar y está declarada en el archivo script.js. Estos son los requerimientos:
  - Si el usuario presiona el botón +1 se debe sumar 1 al valor y se debe modificar el valor en el elemento input.
  - Si el usuario presiona el botón -1 se debe restar 1 al valor y se debe modificar el valor en el elemento input.

¿Preguntas?