/INTRODUCCIÓN A JAVASCRIPT

Curso de FrontEnd

Sábado 26 de Octubre 2024

/AGENDA

/01 /bienvenida

/02 /revisión tarea n 3

/03 /¿Qué es JS?

/04 /Historia (Jquery / AJAX / Cronología)

/05 /¿Cómo usar JS?

/06 /Tipos de datos

/07 /Variables

/08 /Funciones

/09 /tarea n 4

¿Qué es JS?

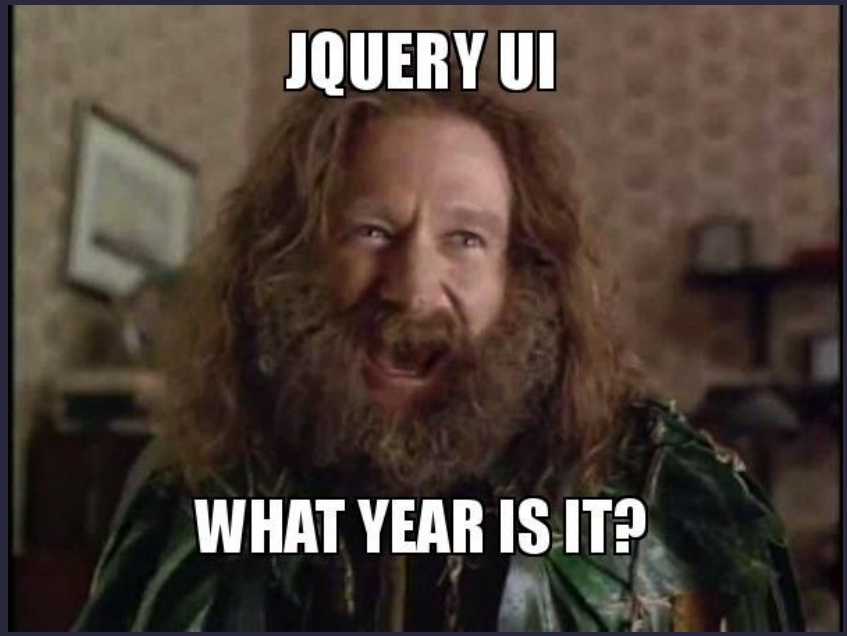
JavaScript (JS) es un lenguaje de programación ligero, interpretado, o compilado justo-a-tiempo (just-in-time) con funciones de primera clase. Si bien es más conocido como un lenguaje de scripting (secuencias de comandos) para páginas web, y es usado en muchos entornos fuera del navegador, tal como Node.js, JavaScript es un lenguaje de programación basada en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa (por ejemplo programación funcional).

- https://developer.mozilla.org/es/docs/Web/JavaScript

Historia: Jquery / Ajax / Cronología JS

jQuery es una librería desarrollada en 2006 por John Resig que permite añadir una capa de interacción AJAX entre la web y las aplicaciones que desarrollemos controlando eventos, creando animaciones y diferentes efectos para enriquecer la experiencia de usuario.

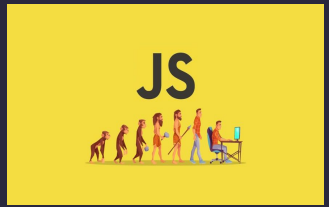
***Ajax** permite que un usuario de la aplicación web interactúe con una página web sin la interrupción que implica volver a cargar la página web



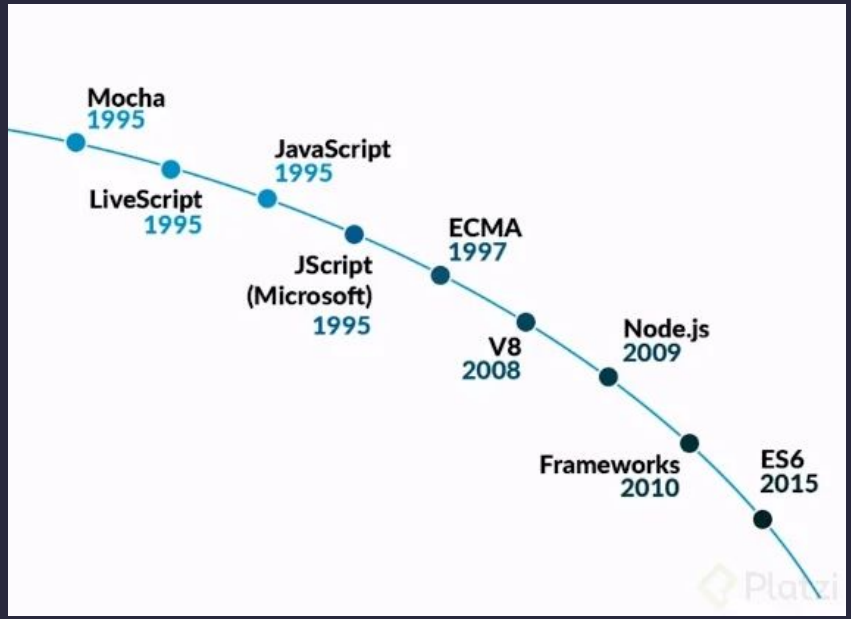
¿Sabías qué...

Javascript fue desarrollado en 10 días? 😲

La primera versión que se lanzó con el nombre de Mocha sí tardó 10 días en desarrollarse. Eso es porque *Netscape* presionó a Brendan en tener una primera versión funcional lo antes posible para poder enfrentarse al empuje de Microsoft para usar Visual Basic en la web.



Historia: Jquery / Ajax / Cronología JS



¿Cómo usar JS?

Ejemplo de script en línea

```
<html>
  <head>
    <title>Título de la página</title>
    <script>
      console.log("¡Hola!");
    </script>
  </head>
  <body>
    <p>Ejemplo de texto.</p>
  </body>
</html>
```

Ejemplo de script externo

```
<html>
  <head>
    <title>Título de la página</title>
    <script src="js/index.js"></script>
  </head>
  <body>
    <p>Ejemplo de texto.</p>
  </body>
</html>
```

https://lenguajejs.com/javascript/introduccion/como-funciona/

Tipos de datos en JS

Tipado dinámico

JavaScript es un lenguaje **débilmente tipado y dinámico**. Las variables en JavaScript no están asociadas directamente con ningún tipo de valor en particular, y a cualquier variable se le puede asignar (y reasignar) valores de todos los tipos:

```
let foo = 42;    // foo ahora es un número
foo = 'bar'; // foo ahora es un string
foo = true;  // foo ahora es un booleano
```

Tipos de datos en JS

Undefined: Una variable a la que no se le ha asignado un valor tiene el valor de undefined.

Boolean: Representa una entidad lógica y puede tener dos valores: true y false.

Number: Variable de tipo número.

Null: Tipo primitivo especial que tiene un uso adicional para su valor: si el objeto no se hereda, se muestra null.

Object: Tipo estructural especial que no es de datos pero para cualquier instancia de objeto construido que también se utiliza como estructuras de datos: new Object, new Array, new Map, new Set, new WeakMap, new WeakSet, new Date y casi todo lo hecho con la palabra clave new

Function: Esta simplemente es una forma abreviada para funciones, aunque cada constructor de funciones se deriva del constructor Object

Variables en JS

Las variables son un concepto fundamental en cualquier lenguaje de programación. En JavaScript, puedes declarar variables usando las palabras clave **var, const o let.** Ejemplo:

```
let edad = 4
```

Variables en JS

¿Qué es el scope en JavaScript?

El **scope** puede definirse como **el alcance que una variable tendrá en tu código**. En otras palabras, el scope **decide a qué variables tienes acceso** en cada parte del código.

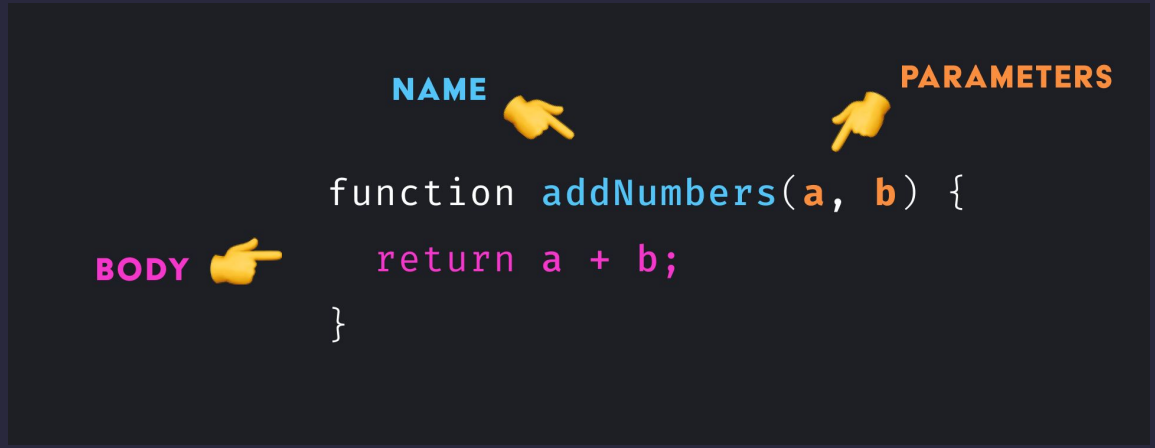
Qué es el Scope Local

Cuando puedes acceder a una variable únicamente en cierta parte del código, se dice que esa variable está declarada en el scope local.

Qué es el Scope Global

Se dice que una variable está en el scope global cuando está declarada fuera de una función o de un bloque.

Funciones en JS



Una función en JavaScript es similar a un procedimiento — un conjunto de instrucciones que realiza una tarea o calcula un valor, pero para que un procedimiento califique como función, debe tomar alguna entrada y devolver una salida.

Para usar una función, debes definirla en algún lugar del ámbito desde el que deseas llamarla.

Funciones en JS

Declaración de función

Una definición de función (también denominada declaración de función o expresión de función) consta de la palabra clave function, seguida de

- El nombre de la función.
- Una lista de parámetros de la función, entre paréntesis y separados por comas.
- Las declaraciones de JavaScript que definen la función, encerradas entre llaves, { ... }.

```
function square(number) {
  return number * number;
}
```

Llamar una función

Definir una función no la ejecuta. Definirla simplemente nombra la función y especifica qué hacer cuando se llama a la función.

Llamar a la función en realidad lleva a cabo las acciones especificadas con los parámetros indicados. Por ejemplo, si defines la función square, podrías llamarla de la siguiente manera:

```
square(5);
```

Tarea Nº4 (opcional)

Desplegar la tarea 3 en GitHub Pages

Crear un repositorio en GitHub y enviarlo mediante formulario:
https://www.softwarelibrechile.cl/G23-S1-04-Tarea

Tarea Nº5

Hacer las funciones que sean necesarias para:

Obtener el promedio de notas de un alumno considerando que la suma de notas debe ser el retorno de una función y el promedio el retorno de otra función. Las notas son: 6,8,9,2,5,10.

Crear un repositorio en GitHub y enviarlo mediante formulario: https://www.softwarelibrechile.cl/G23-S1-05-Tarea

Canales de comunicación

Slack (principal)
www.softwarelibrechile.cl/slack

WhatsApp
www.softwarelibrechile.cl/whatsapp

Programas

Visual Studio Code
www.softwarelibrechile.cl/vscode

Trello
www.trello.com

Codesandbox
codesandbox.io

Programa

01 - Fundamentos
02 - Estilos y diagramación
03 - Estilos y diagramación
04 - Introducción a JS
05 - Primeros pasos a JS
06 - Funciones, API’s, Manejo de errores
07 - Programación JS
08 - ReactJS
09 a 12 - Práctico grupal

Curso FrontEnd: Contenidos

CLASE	CAPÍTULO	CONTENIDOS
SÁBADO 1	**FUNDAMENTOS**	1. BIENVENIDA AGI 2. ORGANIZACIÓN DE TAREAS 3. TRELLO 4. TOMA DE REQUERIMIENTOS BASE 5. METODOLOGÍAS DE TRABAJO 6. HTML, CSS, JS 7. BOOTSTRAP INSPECTOR ELEMENTOS. 8. TAREA N° 1

Curso FrontEnd: Contenidos

SÁBADO 2	**ESTILOS Y DIAGRAMACIÓN CSS PARTE 1**	1. REVISIÓN TAREA N°1 2. BEM, SASS, FLEXBOX 3. PREFIJOS, NAVEGADOR, MEDIAQUERYS 4. Posicionamiento, layout 5. Cajas (tipos, posición) 6. Responsividad 7. Tarea N° 2
SÁBADO 3	**ESTILOS Y DIAGRAMACIÓN CSS PARTE 2**	1. Revisión Tarea N° 2 2. GRID 3. Introducción a GIT 4. Visionamiento de código (GITHUB) 5. Despliegue (GITHUB Pages) 6. Tarea N° 3

Curso FrontEnd: Contenidos

SÁBADO 4	**INTRODUCCION JS**	1. Revisión Tarea N° 3 2. ¿Qué es JS? 3. ¿Cómo usar JS?) 4. JQUERY, AJAX 5. Tipos de datos 6. Funciones 7. Tarea N° 4
SÁBADO 5	**PRIMEROS PASOS CON JS**	1. Revisión Tarea N° 4 2. ARRAYS, CALLBACK 3. Ciclos 4. Tarea N° 5

Curso FrontEnd: Contenidos

SÁBADO 6	**FUNCIONES, API'S, MANEJO DE ERRORES**	1. Revisión Tarea N° 5 2. Tarea N° 6
SÁBADO 7	**PROGRAMACIÓN JS**	1. Revisión Tarea N° 6 2. herencias 3. Tarea N° 7
SÁBADO 8	**INTRODUCCIÓN A REACTJS**	1. Revisión Tarea N° 7 2. Definición proyecto final 3. Tarea N° 8
SÁBADO 9	**PRÁCTICO (GRUPAL)**	1. Revisión Tarea N° 8 2. Proyecto final} 3. Tarea N° 9

Curso FrontEnd: Contenidos

SÁBADO 10	**PRÁCTICO (GRUPAL)**	1. Revisión Tarea N° 9 2. Presentación y entrega
SÁBADO 11	**PROYECTO FINAL (GRUPAL)**	1. Presentación y entrega
SÁBADO 12	**ENTREGA CERTIFICADOS APROBACIÓN**	1. Presentación y entrega 2. Entrega Certificados 3. Cierre

Tarea Nº4

Objetivo:
El propósito de esta tarea es que, en **equipos de dos personas**, trabajen juntos en un proyecto utilizando Git. Deberán aplicar los comandos básicos de Git (clone, push, pull, merge) y trabajar en diferentes ramas para luego combinarlas en la rama principal (main).

Instrucciones:

1. **Creen un repositorio en GitHub:** Uno de los integrantes deberá crear el repositorio y agregar al compañero como colaborador.
2. **Clonen el repositorio:** Ambos deberán clonar el repositorio en sus computadoras locales.
3. **Cree una rama por cada integrante:** Cada uno de los integrantes debe crear su propia rama a partir de main (la rama principal). Nombrar la rama con su nombre o un identificador claro (por ejemplo: rama-nombre1 y rama-nombre2).
4. **Realicen cambios en sus respectivas ramas:** Cada integrante deberá hacer al menos 3 commits en su propia rama, modificando o agregando archivos al proyecto. Pueden agregar contenido como un archivo .md con descripciones, o editar archivos existentes.

5. Hagan push a sus ramas: Una vez que hayan realizado cambios y hecho commits, deberán hacer git push de sus ramas al repositorio remoto en GitHub.
6. Clonen el repositorio: Ambos deberán clonar el repositorio en sus computadoras locales.
6. Combinación de ramas (Merge):

- Una vez que ambos hayan terminado de trabajar en sus ramas, deberán combinar (merge) sus ramas en la rama main.
- Primero, uno de los integrantes debe crear un pull request desde su rama hacia main y hacer el merge.
- Luego, el otro integrante deberá hacer lo mismo con su rama.

7. Resolución de conflictos (si los hay): Si al intentar combinar las ramas surge un conflicto, deberán resolverlo entre ambos. Esto puede implicar que se coordinen para revisar los archivos en conflicto y hacer ajustes.
8. Entrega: Una vez completada la tarea, deben mandar la URL del repositorio en GitHub donde podamos revisar:

- Los commits realizados por ambos.
- Las ramas creadas.
- La combinación de ramas (merge) en main.

Enviar desde el formulario de www.softwarelibrechile.cl/G24-S2-03-Tarea
Fecha de entrega: Martes 05 Noviembre 23:59 hrs

Rúbrica Tarea Nº4

Rúbrica Tarea Nª4 - Evaluación para Tarea de Colaboración en Git

Suma el porcentaje alcanzado en cada criterio para obtener una nota del 0 al 10.

Esta rúbrica refleja la colaboración en equipo usando Git, la importancia de seguir los pasos y aplicar los comandos necesarios, y la correcta documentación de los cambios y entregas.

Criterio	Descripción	Ponderación	Notas
Creación del Repositorio y Colaboración	Un integrante creó el repositorio en GitHub y agregó al compañero como colaborador correctamente.	15%	0: No se creó el repositorio o no se agregó al colaborador. 5-9: Repositorio creado, pero sin agregar colaborador o con errores. 10-14: Repositorio y colaboración correctamente realizados con detalles menores. 15: Creación impecable.
Clonación del Repositorio	Ambos integrantes clonaron correctamente el repositorio en sus máquinas locales.	10%	0: No se clona el repositorio. 5: Clonación incorrecta o incompleta. 6-9: Clonación realizada pero con algún error. 10: Clonación exitosa por ambos integrantes.
Creación y Nomenclatura de Ramas	Cada integrante creó una rama con un nombre claro y coherente a partir de main.	15%	0: No se crean ramas o están mal nombradas. 5-9: Ramas creadas pero con errores en el nombre o en su creación. 10-14: Ramas bien creadas, pero algún error menor. 15: Ramas correctamente creadas y nombradas.
Commits en Ramas	Cada integrante hizo al menos 3 commits en su rama, con mensajes de commit claros y significativos.	20%	0: No se realizaron commits. 5-9: Menos de 3 commits por integrante o commits con mensajes confusos. 10-14: Se realizaron 3 commits, pero los mensajes no son claros. 15-19: Commits realizados con mensajes adecuados. 20: Se hicieron los commits correctos con mensajes significativos.
Push de Ramas al Repositorio Remoto	Ambos integrantes hicieron git push de sus ramas al repositorio en GitHub.	10%	0: No se hizo el push. 5: Push incompleto o incorrecto. 6-9: Push realizado con algún detalle a mejorar. 10: Push exitoso de ambas ramas.
Combinación de Ramas (Merge)	Se realizaron correctamente los pull requests y el merge de las ramas en main.	15%	0: No se realizó ningún merge. 5-9: Merge realizado pero con errores importantes. 10-14: Merge realizado con pequeños errores. 15: Merge realizado correctamente sin errores.
Resolución de Conflictos (si aplica)	En caso de conflictos, estos fueron resueltos correctamente entre los integrantes.	10%	0: No se resolvieron los conflictos. 5: Conflictos resueltos, pero de manera incorrecta. 6-9: Conflictos resueltos con algunos errores. 10: Conflictos resueltos de manera efectiva.
Entrega del Repositorio en GitHub	Entrega correcta del enlace al repositorio donde se visualizan todas las ramas, commits y merges.	5%	0: No se entregó el repositorio. 3: Se entregó el repositorio pero con información incompleta. 5: Entrega completa y correcta.

Canales de comunicación

Slack (principal)
www.softwarelibrechile.cl/slack

WhatsApp
www.softwarelibrechile.cl/whatsapp

/Porcentajes a cumplir en el curso

A continuación les dejamos información acorde al % de asistencia, entregas de tarea y proyecto final, todo esto de **carácter obligatorio** para obtener el certificado de aprobación.

/CLASES	/TAREAS	/PROYECTO FINAL
ASISTENCIA 80%	**ENTREGAS 80%**	**ENTREGA 100%**
11 clases	8 tareas	1 proyecto
Debe existir asistencia mínima de 9 clases	Deben entregarse mínimo 7 tareas o más	El proyecto final se hará de forma grupal, 2 personas, ambas obteniendo la misma nota.

UNIVERSIDAD DE SANTIAGO DE CHILE

/APROBACIÓN Y ENTREGA CERTIFICADO

/TAREAS

El promedio de las notas de la tareas equivale a un

/50%

+

/PROYECTO FINAL

La nota en el proyecto final equivale a un

/50%

=

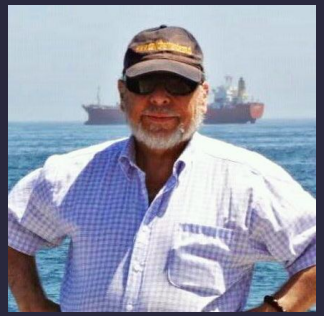
/NOTA FINAL

La nota debe ser igual o mayor a 6

/100%

UNIVERSIDAD DE SANTIAGO DE CHILE

Curso FrontEnd: Relatores



David Hernández

Director Asociación de Informáticos UTE-USACH A.G.|Paleo Informático
david.hernandezm@usach.cl
+56998246832

https://www.linkedin.com/in/david-hern%C3%A1ndez-23205b3a

Gonzalo Flemming

Tech Lead en Falabella Financiero

https://www.linkedin.com/in/gfleming-garrido/

Sebastian Becerra

Gerente de Operaciones en PropulsoW, Ingeniero Informático, voluntario Mozilla Chile ayudante en taller Joomla dictado en la Usach el 2014- 2015

https://www.linkedin.com/in/sebaebc/

Scarlet Melgarejo

Encargada de Canales Digitales B2B - B2C en Productos Torre , Ayudante en Desafío Latam, voluntario Mozilla Chile ayudante en taller Joomla dictado en la Usach el 2011 - 2015 2015.

https://www.linkedin.com/in/scarlett-melgarejo-venegas-38805626/

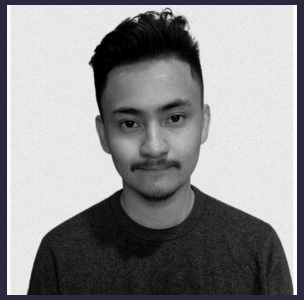
UNIVERSIDAD DE SANTIAGO DE CHILE

Curso FrontEnd: Relatores

Carolina Pirela

Ingeniera en Sistemas - Front End Developer React | Javascript | Sass | Boostrap
Dev UI Mobile @ Seiza

https://www.linkedin.com/in/caropi30/



Cristian Pavéz

Head of AI and Data Innovation en Soluciones - an AVOS Tech Company
Ingeniero en Informática

https://www.linkedin.com/in/cristian-pavez

<gracias!>