Unidad 1

Resumen Unidad 1: Selección de arquitecturas y herramientas de programación

1. Introducción al desarrollo web

El desarrollo web se divide en **dos partes principales**:

Back-end (lado del servidor)

- Parte no visible de la web
- Ejemplos: bases de datos, scripts que se ejecutan en el servidor

Front-end (lado del cliente)

- Parte visible de la web
- Ejemplos: código HTML, CSS, scripts que se ejecutan en el lado del cliente

Diferencias clave entre cliente y servidor

Aspecto	Lado Cliente	Lado Servidor
Definición	Implementación efectiva de los componentes visuales de una aplicación web	Implementación efectiva de funcionalidades de una aplicación web, donde se incluyen el acceso a datos, la administración de servidores, etcétera
Habilidades requeridas	HTML, CSS, SASS, JavaScript...	Python, Ruby, Java, PHP...
Independencia	No puede funcionar de forma independiente excepto en el caso de sitios estáticos	Funciona de forma independiente respecto del front-end

Aspecto	Lado Cliente	Lado Servidor
Frameworks utilizados	AngularJS, React, vue.js...	Django, Flask, CakePHP, Laravel, Ruby on Rails...

2. Framework

> **Framework**: conjunto de herramientas y librerías software que proporcionan numerosas utilidades para el desarrollo de aplicaciones web más escalables y sencillas de mantener, proporcionan un importante ahorro de recursos.

Ventajas de los frameworks:

- **El coste**: Muchos de estos frameworks son de código abierto, con lo cual no hay que realizar ninguna inversión
- **Están suficientemente probados** y su código suele carecer de errores, puesto que muchos programadores lo utilizan. Además, suelen tener un alto nivel de seguridad y rendimiento
- **Permite desarrollar mucho más rápido**, puesto que muchas de las estructuras, clases, patrones de diseño, etc., vienen ya incorporadas en el framework
- **Cualquier persona que maneje un framework determinado puede entender** e incorporarse de forma eficiente y efectiva a un equipo de desarrollo que lo esté utilizando en un proyecto determinado

3. Diseño UX y UI

UX (User eXperience) - Experiencia de usuario

- Busca una experiencia amigable, sencilla y lo más intuitiva posible

UI (User Interface) - Interfaz de usuario

- Determina la interacción del usuario con la aplicación a través de los elementos diseñados para ello

Ambas determinan el grado de satisfacción de uso de una aplicación.

Componentes del diseño

UX incluye:

- Diseño de Interacción
- Prototipos
- Arquitecto de Información
- Investigación de usuarios
- Escenarios

UI incluye:

- Diseño Visual
- Colores
- Diseñador Gráfico
- Diseños
- Tipografía

---

4. Front-end (lado cliente)

En la parte front-end prima la parte creativa y la originalidad. Es el perfil más cercano al diseñador, aunque también se trabaja en el código.

Lenguajes principales:

HTML

- No es un lenguaje de programación sino un lenguaje de marcas
- Define el contenido que va a tener el documento
- La función del navegador web es leer e interpretar todo el contenido, junto con las etiquetas y mostrárselo al cliente
- **Ventaja**: cualquier navegador debería visualizar el contenido de la página web de la misma forma y con el mismo aspecto

CSS

- Define la presentación del documento
- Es un lenguaje de diseño gráfico
- Su objetivo es que la página web sea atractiva al usuario
- No modifica el comportamiento ni el contenido, sino sólo el aspecto de la página web

JavaScript

- Agrega contenido dinámico a las páginas web
- Es un verdadero lenguaje de programación

5. Scripts vs Lenguajes de Programación

Característica	Scripts	Lenguajes de Programación
Compilación	Lenguajes interpretados, no necesitan ser compilados	Se suelen compilar
Componentes	Utilizan componentes ya preexistentes	A veces, se empiezan a desarrollar desde 0
Ejecución	Se suelen incrustar dentro de otros programas	Se pueden ejecutar de forma independiente
Ejecución línea a línea	Se ejecutan línea a línea. Pueden producirse muchos errores en ejecución	Se pueden ejecutar en bloques
Fichero ejecutable	No generan un fichero ejecutable	Generan un fichero ejecutable
Facilidad de uso	Diseñados para ser fáciles de utilizar y programar	No siempre son fáciles de utilizar y programar
Ejemplos	JavaScript, Shell, Perl, PHP, Python, Ruby, etc.	Java, C, C++, Swift, Pascal, etc.

6. Introducción a JavaScript

¿Por qué elegir JavaScript?

- Es muy fácil de implementar: tan solo es necesario colocar el código en un documento HTML y decirle al navegador que es JavaScript
- Funciona en todos los navegadores que usan los usuarios de la web, incluso cuando están desconectados
- Permite crear interfaces altamente amigables que mejoran la experiencia del usuario y brindan mucho dinamismo, sin tener que esperar a que el servidor reaccione y muestre otra página

- Puede cargar contenido de forma asíncrona en el documento si el usuario lo necesita y cuando lo necesite, sin recargar toda la página
- Puede comprobar lo que es posible hacer en un navegador y reaccionar en consecuencia
- Puede ayudar a solucionar problemas de incompatibilidades entre navegadores y corregir problemas de diseño con CSS

Frameworks de JavaScript más utilizados:

ReactJS

- Framework creado por Facebook
- Permite realizar aplicaciones web de una forma rápida y eficiente
- Renderiza (dibuja) los componentes del front-end de una forma sencilla y eficaz
- Utiliza programación orientada a objetos

AngularJS

- Framework creado y mantenido por Google
- Ha pasado de ser una librería a ser una plataforma de desarrollo
- No es fácil de aprender
- Actualmente se programa en TypeScript, superconjunto de JavaScript

Vue.js

- Framework de gran ligereza y velocidad de ejecución
- Es algo más sencillo de aprender que Angular

> 💡 **Nota**: Actualmente, las empresas no programan en JavaScript puro, sino que se basan en un framework de este.

---

7. Herramientas para el desarrollo en JavaScript

1. NAVEGADOR

Sirve para ejecutar las aplicaciones en su contexto. Según el porcentaje de uso (junio 2022):

- Chrome: 65,87%
- Safari: 18,62%
- Edge: 4,12%
- Firefox: 3,26%
- Samsung: 2,88%
- Opera: 2,12%

Recomendaciones:

- Los desarrolladores deben trabajar con varios navegadores
- Se recomienda probar para los 4 navegadores con mayor cuota de mercado
- Optimizar todo lo posible para el navegador dominante
- Probar diferentes configuraciones del navegador para ajustarlo al mayor número de tamaños de pantalla posibles (usando plugins)

2. EDITOR DE CÓDIGO

Herramienta que se utiliza para escribir el código e incluyen características que facilitan el trabajo:

- Comprobación de errores
- Autocompletado
- Resaltado de la sintaxis

Tipos de editores:

- **Editores de código**: Atom, Sublime Text, Brackets, Coda...
- **Entornos de desarrollo integrados (IDE)**: Visual Studio, Eclipse, Netbeans, Webstorm...
- **Editores online**: CodePen, JSFiddle, JS Bin...

💡 La mejor forma de elegir un editor de código es probar todos los que se pueda. No es necesario centrarse en un único editor.

3. INTÉRPRETE DE JAVASCRIPT

Por la naturaleza de JavaScript no es necesario realizar ninguna instalación adicional para el intérprete ya que viene integrado en los navegadores web.

Verificar que esté habilitado:

- **Chrome**: Configuración → Seguridad y privacidad → Configuración de sitios → JavaScript → Los sitios pueden usar JavaScript
- **Safari**: Herramientas → Preferencias → Seguridad → Activar JavaScript
- **Edge**: Opciones → Mostrar opciones avanzadas → Privacidad → Configuración de contenido → Permitir que todos los sitios ejecuten JavaScript (recomendado)
- **Firefox**: Herramientas → Opciones → Contenido → Activar JavaScript

---

8. Herramientas de programación en JavaScript

Existen múltiples alternativas:

1. **Herramientas online**: https://www.tutorialspoint.com/codingground.htm
2. **IDE con sistema de control de versiones**:
   - ATOM: https://github.com/atom
   - SUBLIME TEXT: https://www.sublimetext.com/
   - VISUAL STUDIO CODE: https://code.visualstudio.com/

---

9. Integración de JavaScript en HTML

JavaScript se complementa al código HTML de una página web.

Dos opciones:

1. **Integrar el código JavaScript dentro de los archivos HTML**

```
<script>
  // código JavaScript aquí
</script>
```

2. **Tener separado del HTML el código JavaScript en archivos con extensión .js**

```
<script src="script.js"></script>
```

> ✅ **Mejor práctica**: Lo más limpio y eficaz es tener el código JavaScript fuera de los archivos HTML por las ventajas que ofrece: las páginas cargarán mucho más rápido, se independiza el HTML del código y el JavaScript será mucho más fácil de mantener.

Organización recomendada:

Los proyectos suelen tener los scripts en una carpeta aparte de nombre **script** o **js** para tener los archivos más ordenados.

---

10. Ejemplos de JavaScript

1. Modificación del contenido de una página web

```
document.getElementById('prueba').innerHTML = 'CAMBIANDO el contenido!'
```

2. Cambiar atributos de objetos HTML

```
var image = document.getElementById('imgJS')
if (image.src.match("js")) {
    image.src = "img/html.png";
} else {
    image.src = "img/js.png";
}
```

3. Cambiar el estilo CSS

```
var x = document.getElementById("mytxt");
x.style.fontSize = "25px";
x.style.color = "red";
```

---

11. Comunicación de JavaScript con el exterior

Existen varias formas para que el código JS se comunique con el usuario:

1. **console.log()** - Escribir en la consola del navegador

Esta opción solo la pueden utilizar los desarrolladores.

```
console.log("Cesur, ejemplo de consola");
```

Para acceder a la consola, basta con pulsar F12 y elegir Consola en su navegador.

2. **innerHTML** - Escribir en el HTML

```
document.getElementById("texto").innerHTML = 5 + 6;
```

3. **document.write()** - Generar directamente HTML

```
document.write("<h2>¡¡Hola clase!!</h2>");
```

4. **window.alert()** o **alert()** - Generar un mensaje de alerta

Muestra un diálogo emergente en el navegador

```
window.alert("¡Hola clase!!");
```

Conclusión

Esta unidad introduce las bases del desarrollo web moderno, con especial énfasis en JavaScript como lenguaje de programación del lado del cliente. Se han cubierto los conceptos fundamentales de front-end y back-end, las herramientas necesarias para el desarrollo, y las diferentes formas de integrar y comunicar JavaScript con las páginas web.