Clase 24

Extensiones de navegador

IIC2513 - Tecnologías y Aplicaciones Web

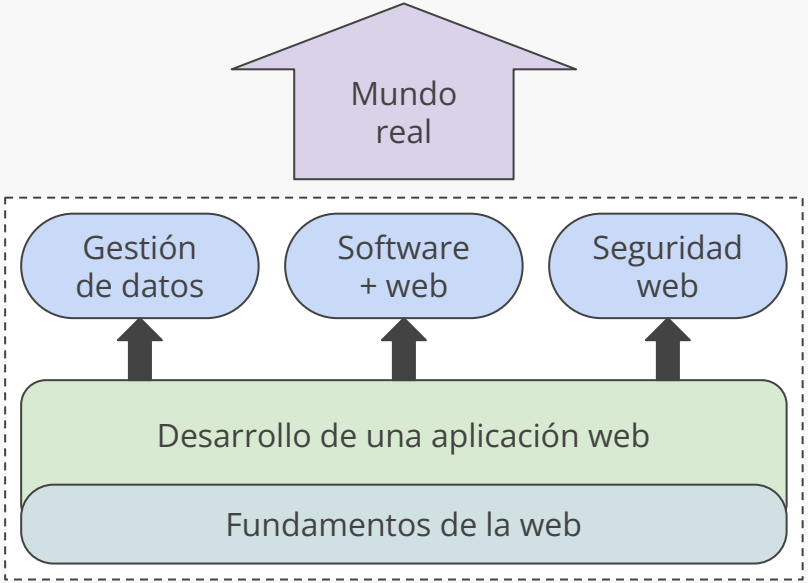
Antonio Ossa Guerra
aaossa@ing.puc.cl

1. Mucho ánimo con lo que queda
2. Recuerden que existe la ECA
3. Solo 3 clases más

Anuncios

Contenidos del curso

(1) Fundamentos de la web
(2) Desarrollo de una app web
(3) Gestión de datos
(4) Software + web
(5) Seguridad web
(6) El mundo real

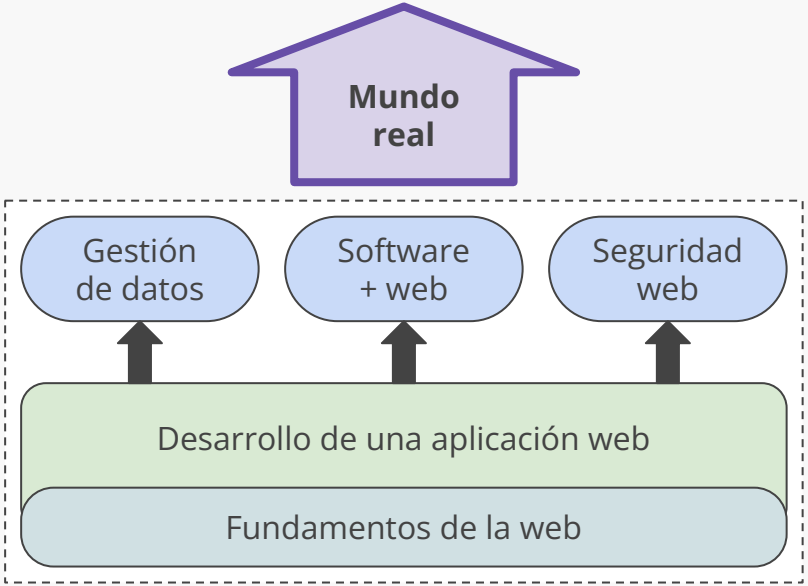


Contenidos del curso

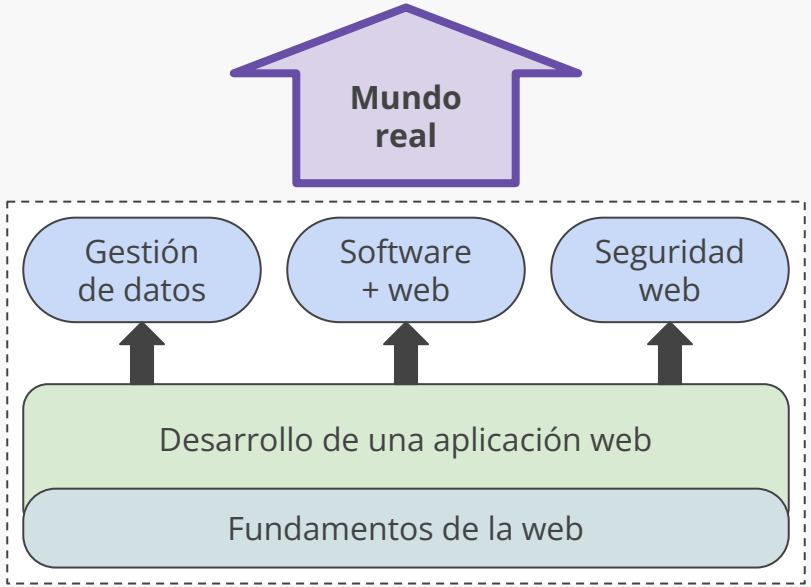
El mundo real

Discutiremos temas como **privacidad**, **ética** y **cómo son el desarrollo y las aplicaciones web en el mundo real**...

... para asegurar de que los aprendizajes del curso permiten crear **soluciones reales, útiles y que generen valor** para la sociedad



Contenidos del curso



Menti#*Extensiones de navegador*

Agenda

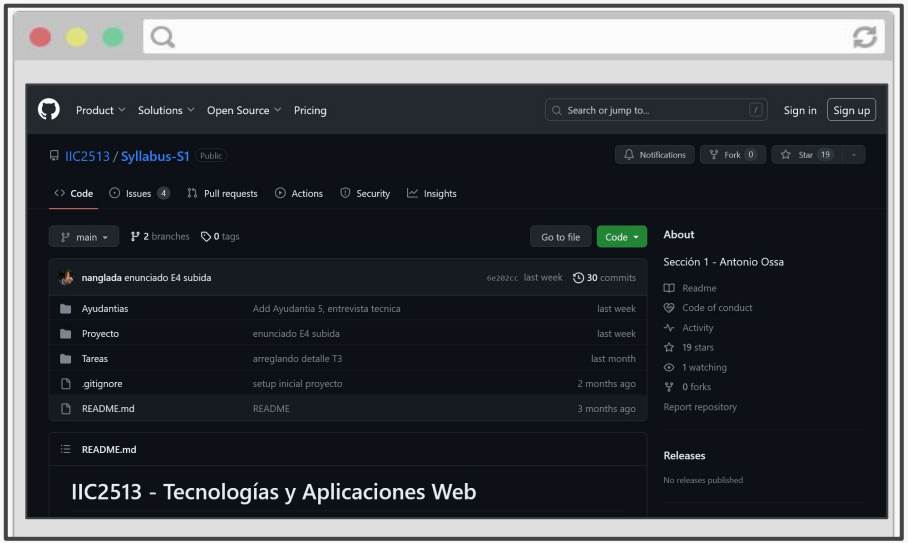
✨ **Qué son**

Hello world!

Hagamos una extensión

Extensiones de navegador

Software que extiende la funcionalidad del navegador. Código basado en HTML, CSS y JavaScript, que puede aprovechar las mismas API Web que JavaScript en una página web, además de su propio conjunto de APIs





Sobre navegadores...

Al igual que con los editores de código, un navegador es una herramienta para lograr un objetivo. Si bien el funcionamiento de mecanismo a utilizar está estandarizado, los navegadores implementan y soportan funcionalidades de manera levemente distinta

Lo más importante es **elegir un navegador con el que sientan comodidad y familiaridad** para su uso diario (incluyendo su rol de desarrolladores). Esto incluye desde el uso básico hasta *keyboard shortcuts* y customizaciones



Google Chrome



Microsoft Edge



Mozilla Firefox

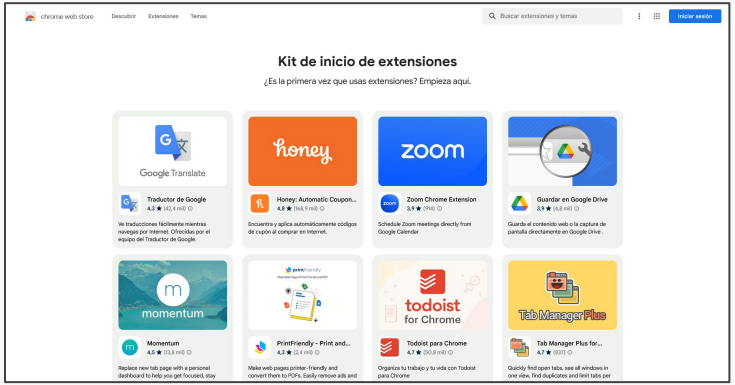


Apple Safari

Extensiones de navegador

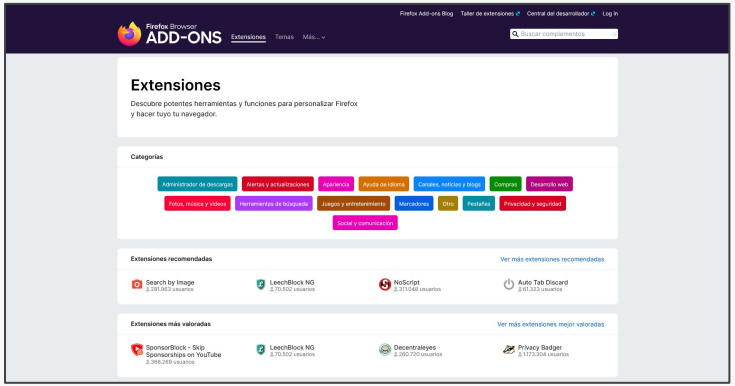
Chrome:

[Chrome Web Store - Kit básico de extensiones]



Firefox:

[Extensiones - Complementos para Firefox]



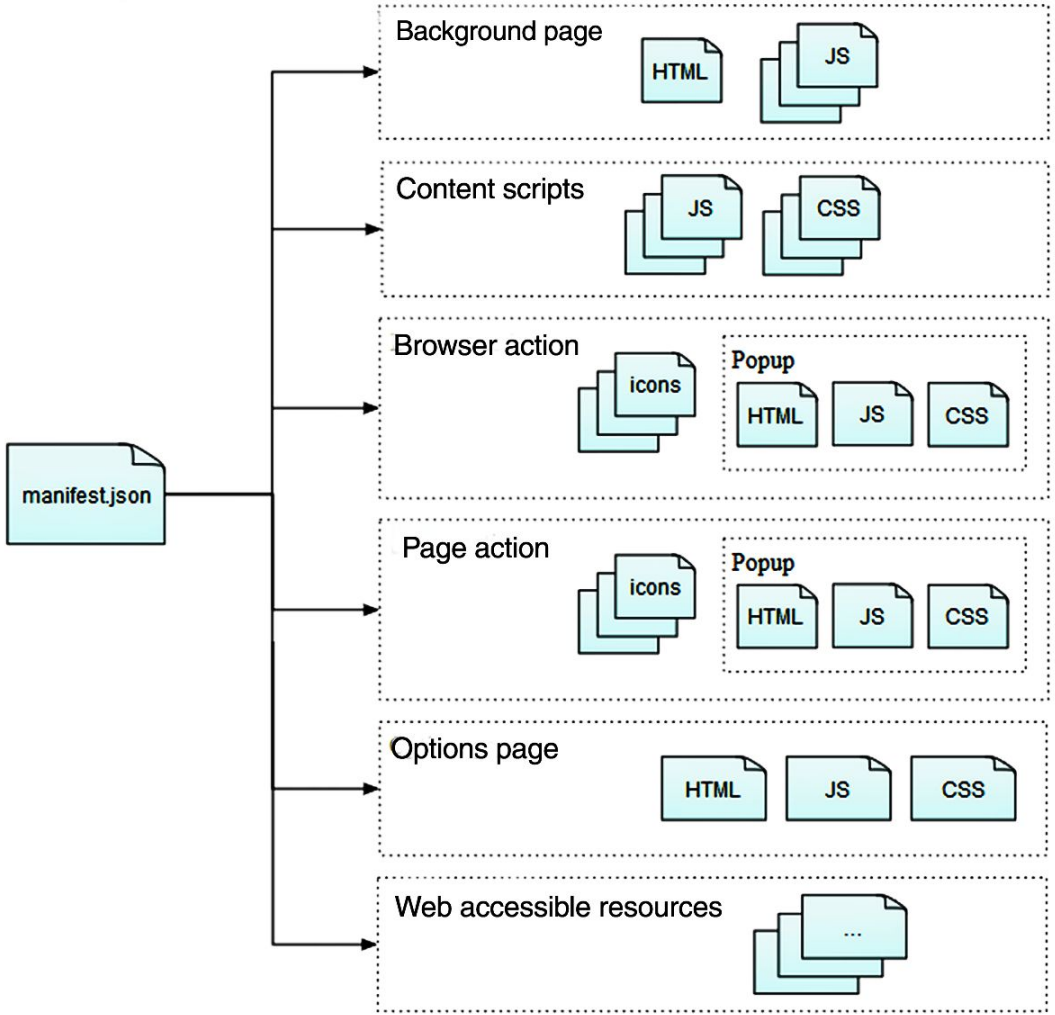
Arquitectura

Extension architecture basics (10 min)

- Content scripts vs background scripts
- Popup interfaces and options pages
- Quick manifest.json overview

Una extensión es un conjunto de archivos, empaquetados para su distribución e instalación.

Fuente: *Anatomy of an extension - Mozilla | MDN*



Agenda

Qué son

✨ **Hello world!**

Hagamos una extensión

¿Cómo probar la extensión en mi navegador?

Firefox

1. Ir a `about:debugging`
2. Hacer clic en “Cargar complemento temporal”
3. Seleccionar cualquier archivo en el directorio de la extensión

Instrucciones oficiales

Chrome (y otros *chromium-based*)

1. Ir a `chrome://extensions`
2. Activar “Developer mode”
3. Presionar “Load unpacked”
4. Seleccionar la carpeta en la que se encuentra la extensión

Instrucciones oficiales

manifest.json

Los parámetros `manifest_version`, `name` y `version` son **obligatorios**, y contienen metadata básica para la extensión

Si encuentran algún ejemplo con `manifest_version` v2, aquí pueden ver cómo migrar a v3

```
{
  "manifest_version": 3,
  "name": "Hello world",
  "version": "1.0",
  "description": "Says 'Hello world!'",
  "icons": {
    "48": "icons/pizza.svg"
  },
  "content_scripts": [
    {
      "js": ["hello.js"],
      "matches": ["<all_urls>"]
    }
  ]
}
```

Fuente: *manifest.json - Mozilla | MDN*

manifest.json

El parámetro `description` es opcional pero recomendado, actuando como una breve descripción de la extensión (en Firefox y Chrome el máximo es de 132 caracteres)

```
{
  "manifest_version": 3,
  "name": "Hello world",
  "version": "1.0",
  "description": "Says 'Hello world!'",
  "icons": {
    "48": "icons/pizza.svg"
  },
  "content_scripts": [
    {
      "js": ["hello.js"],
      "matches": ["<all_urls>"]
    }
  ]
}
```

Fuente: *description - Mozilla | MDN*

manifest.json

`icons` es otro parámetro opcional, y permite especificar un ícono para la extensión.

Podemos indicar distintos tamaños (48 es recomendado) y usar formatos como png o svg

```
{
  "manifest_version": 3,
  "name": "Hello world",
  "version": "1.0",
  "description": "Says 'Hello world!'",
  "icons": {
    "48": "icons/pizza.svg"
  },
  "content_scripts": [
    {
      "js": ["hello.js"],
      "matches": ["<all_urls>"]
    }
  ]
}
```

Fuente: *icons - Mozilla | MDN*

manifest.json

Finalmente, el parámetro `content_scripts` indica cuáles scripts (`js`) se cargarán en las páginas web indicadas (`matches`)

```
{
  "manifest_version": 3,
  "name": "Hello world",
  "version": "1.0",
  "description": "Says 'Hello world!'",
  "icons": {
    "48": "icons/pizza.svg"
  },
  "content_scripts": [
    {
      "js": ["hello.js"],
      "matches": ["<all_urls>"]
    }
  ]
}
```

Fuente: *content_scripts - Mozilla | MDN*

hello.js

Empezaremos con un script simple: imprimir “Hello world!” en la consola una vez que el documento sea cargado.

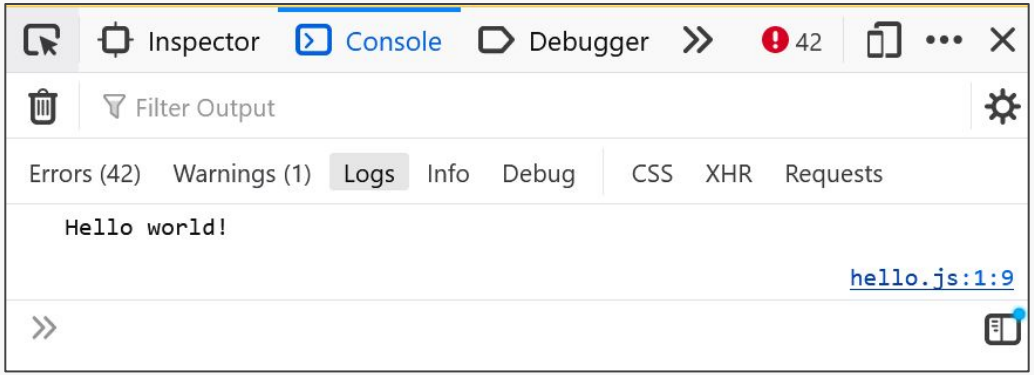
Este archivo, será inyectado después de cualquier archivo `*.css` en el momento indicado en `run_at` (otra key de `content_script`)

```
// hello.js

console.log('Hello world!');
```

Resultado

Cuando la extensión está cargada, este mensaje aparecerá en consola en cualquier página, debido que especificamos que el *script* se ejecute para todas las páginas



Demo*#Vamos al navegador*

Agenda

Qué son

Hello world!

✨ **Hagamos una extensión**

hello.js

Debido a que este *script* se cargará en las páginas que coincidan con el patrón dado en el parámetro `content_scripts`, podemos acceder al documento, de la misma manera que los *scripts* cargados por la propia página

```
// hello.js

function addBorderToAllImages() {
  const images = document.querySelectorAll('img');
  images.forEach(img => {
    img.style.border = '3px solid red';
  });
  console.log(`Added borders to ${images.length} images`);
}

addBorderToAllImages();
```

manifest.json

Debido a que este *script* se cargará en las páginas que coincidan con el patrón dado en el parámetro `content_scripts`, podemos acceder al documento, de la misma manera que los *scripts* cargados por la propia página

```
{
  ...
  "content_scripts": [
    {
      "js": ["hello.js"],
      "matches": ["https://*.wikipedia.org/*"]
    }
  ]
}
```

Resultado

Al activar la extensión, agregamos un borde rojo en toda imagen visible (solo en Wikipedia):



Demo*#Vamos al navegador*

manifest.json

Adicionalmente, podemos especificar que nuestra extensión muestra un *popup*, que en general se utiliza para desplegar opciones y acciones a los usuarios

```
{
  ...
  "permissions": ["activeTab"],
  "action": {
    "default_icon": "icons/pizza.svg",
    "default_title": "Hello",
    "default_popup": "popup/hi.html"
  }
}
```

Fuente: *Your second extension - Mozilla*

hi.html

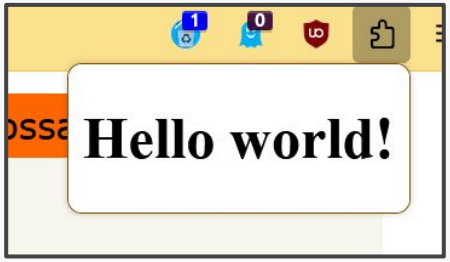
Para especificar el contenido que se muestra al hacer clic en el ícono de la extensión (lo que se muestra en el *popup*), utilizamos html

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
</head>
<body>
    <h1>Hello IIC2513!</h1>
</body>
</html>
```

Fuente: *Popups - Mozilla | MDN*

Resultado

El *popup* aparecerá cada vez que presionemos el ícono de la extensión que creamos:



Demo*#Vamos al navegador*

Recursos adicionales

- Browser extensions - Mozilla | MDN (español): documentación de MDN sobre funcionamiento, implementación (guiada) y ejemplos de extensiones para navegador. Un buen lugar para partir
- Example extensions - Mozilla | MDN: ejemplos de extensiones desarrolladas por MDN, con código abierto, que incluye la lista de Javascript APIs utilizadas en cada uno de los ejemplos
- Extension Workshop: Recursos de Firefox para iniciar en el desarrollo de extensiones
- mozilla/webextension-polyfill (GitHub): Polyfill para extensiones multi-*browser*

Clase 24

Extensiones de navegador

IIC2513 - Tecnologías y Aplicaciones Web

Antonio Ossa Guerra
aaossa@ing.puc.cl