

Pjax and the next generation of server-side Web frameworks

Adrian Holovaty



Web framework





Web framework
returns full page



Web framework
returns full page



Web page - Wikipedia, the free encyclopedia

en.wikipedia.org/wiki/Web_page

Log in / create account

Article Talk

Read Edit View history Search

Web page

From Wikipedia, the free encyclopedia

A **web page** or **webpage** is a document or information resource that is suitable for the World Wide Web and can be accessed through a web browser and displayed on a monitor or mobile device. This information is usually in HTML, or XHTML format, and may provide navigation to other web pages via [hypertext](#) links. Web pages frequently subsume other resources such as style sheets, scripts and images into their final presentation.

Web pages may be retrieved from a local computer or from a remote web server. The web server may restrict access only to a private network, e.g. a corporate intranet, or it may publish pages on the World Wide Web. Web pages are requested and served from web servers using Hypertext Transfer Protocol (HTTP).

Web pages may consist of files of static text and other content stored within the web server's file system (static web pages), or may be constructed by server-side software when they are requested (dynamic web pages). Client-side scripting can make web pages more responsive to user input once on the client browser.

A screenshot of a web page on Wikipedia

Contents [hide]

- 1 Colour, typography, illustration, and interaction
 - 1.1 Dynamic behavior
- 2 Browsers
- 3 Elements
- 4 Rendering
- 5 URL
- 6 Viewing
- 7 Creation
- 8 Saving
- 9 See also
- 10 References

Colour, typography, illustration, and interaction

Web pages usually include information as to the colours of text and backgrounds and very often also contain links to images and sometimes other types of media to be included in the final view. Layout, typographic and color-scheme information is provided by Cascading Style Sheet (CSS) instructions, which

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Toolbox
Print/export

Languages
العربية
Azərbaycanca
Бân-lâm-gú
Беларуская
Беларуская (тарашкевіца)
Български
Boarisch
Català
Česky
Deutsch

en.wikipedia.org/wiki/HTML

Hypertext - Wikipedia, the free encyclopedia

en.wikipedia.org/wiki/Hypertext

Log in / create account

Article Talk

Read Edit View history Search

Hypertext

From Wikipedia, the free encyclopedia

"Metatext" redirects here. For the literary concept, see Metafiction.

Hypertext is text displayed on a computer or other electronic device with references (hyperlinks) to other text that the reader can immediately access, usually by a mouse click or keypress sequence. Apart from running text, hypertext may contain tables, images and other presentational devices. Hypertext is the underlying concept defining the structure of the World Wide Web.^[1] It is an easy-to-use and flexible format to share information over the Internet.

Contents [hide]

- 1 Etymology
- 2 Types and uses of hypertext
- 3 History
- 4 Implementations
- 5 Academic conferences
- 6 Hypertext fiction
 - 6.1 Critics and theorists
- 7 See also
- 8 References
- 9 Further reading
- 10 External links

Etymology

The prefix **hyper-** (comes from the Greek prefix "υπερ-" and means "over" or "beyond", while having common origins with the English word "super") signifies the overcoming of the old linear constraints of written text. The term "hypertext" is often used where the term "hypermedia" might seem appropriate. In 1992, author Ted Nelson – who coined both terms in 1963 – wrote:

By now the word "hypertext" has become generally accepted for branching and responding text, but the corresponding word "hypermedia", meaning complexes of branching and responding graphics, movies and sound – as well as text – is much less used. Instead they use the strange term "interactive multimedia": this is four syllables longer, and does not express the idea of extending hypertext.

– Nelson, *Literary Machines*, 1992

Types and uses of hypertext

Hypertext documents can either be static (prepared and stored in advance) or dynamic (continually changing in response to user input). Static hypertext can

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Toolbox
Print/export

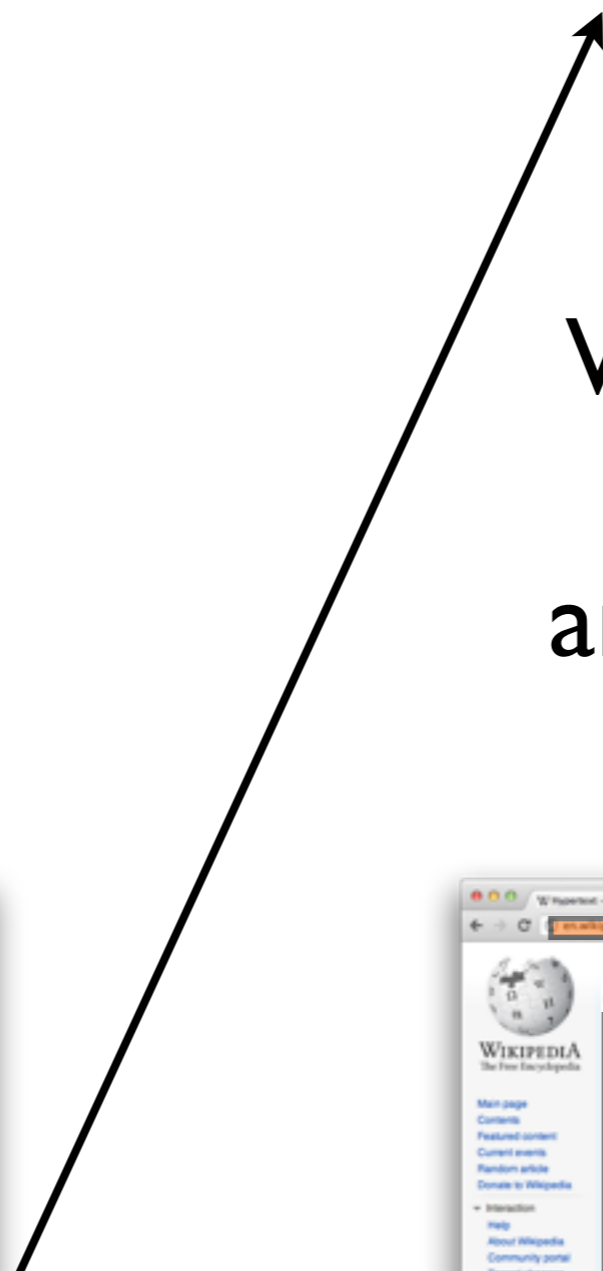
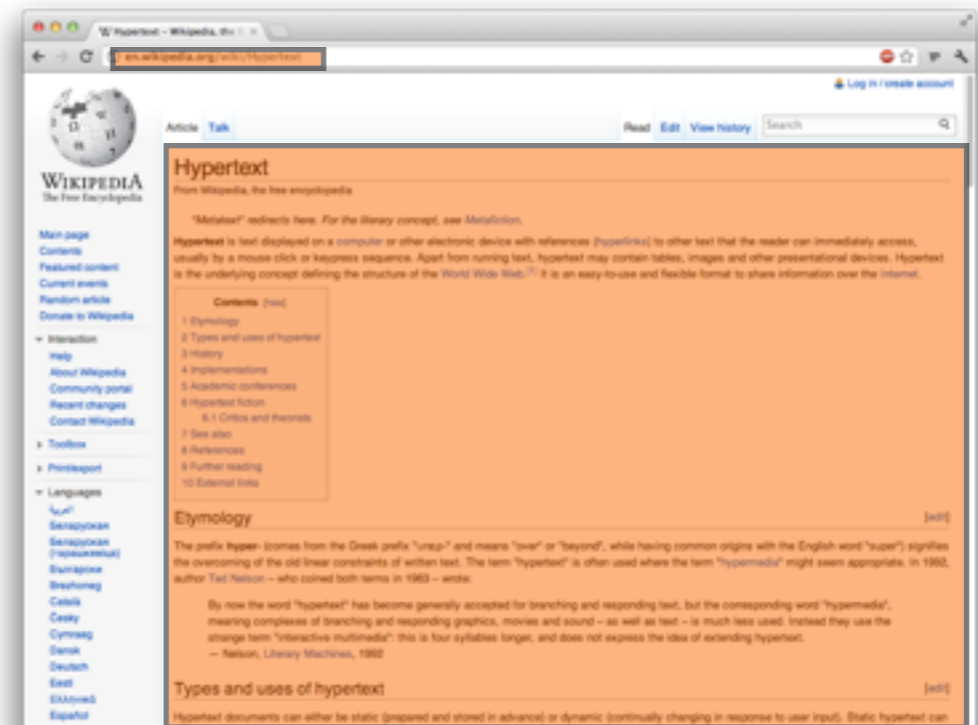
Languages
العربية
Беларуская
Беларуская (тарашкевіца)
Български
Boarisch
Català
Česky
Cymraeg
Dansk
Deutsch
Eesti
Ελληνικά
Español



Web framework returns full page



Web framework returns *diff* and changes URL





Bing and Google Agree: Slow Pages Lose Users

[Print](#) [Listen](#)

by [Brady Forrest](#) | [@brady](#) | [Comments: 22](#) | 23 June 2009

Today representatives of [Google Search](#) and [Microsoft's Bing](#) teams, Jake Brutlag and Eric Schurman respectively, presented the results of user performance tests at today's Velocity Conference. The talk was entitled [The User and Business Impact of Server Delays, Additional Bytes, and HTTP Chunking in Web Search](#). These are long-term tests were designed to see what aspects of performance are most important. To know how to improve their sites both Bing and Google need to know what tweaks to page load perceptions and realities help or hurt the user experience. This is one of the first performance tests that has actual data (and is not strictly anecdotal). The numbers may seem small, but they if you are dealing in millions/billions they add up quickly.

Here are Brutlag's and Schurman's final points:

- "Speed matters" is not just lip service
- Delays under half a second impact business metrics
- The cost of delay increases over time and persists
- Use progressive rendering
- Number of bytes in response is less important than what they are and when they are sent

PJAX

(P = pushstate)



EveryBlock General

Seattle postcard

Posted by Sandor Welsz 23 hours ago

Edit

Delete...



How's this look, Marina?



Seattle.pdf

Discussion



Add a comment...

History

Apr 9 at 10:20am: Sandor W. posted the message (Becca Martin and Marina Gordon were notified)

django/django at master · adrianholovaty · GitHub

GitHub, Inc. [US] <https://github.com/adrianholovaty/django/tree/master/django>

github Search... Explore Gist Blog Help adrianholovaty

adrianholovaty / django
forked from django/django

Admin Unwatch Fork Pull Request 2 747

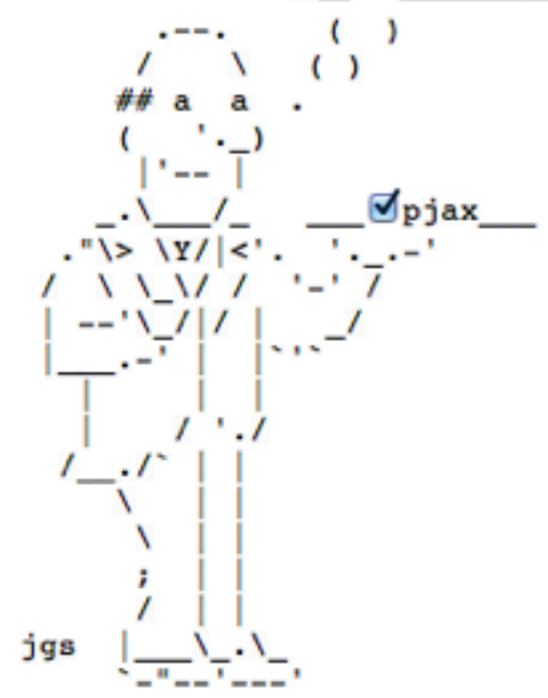
Code Network Pull Requests 0 Stats & Graphs

branch: master Files Commits Branches 16 Tags 36 Downloads

django / django

name	age	message	history
..			
bin	9 months ago	Fixed #16225 – Removed unused imports. Many thanks to Aymeric August... [jezdez]	
conf	10 days ago	Removed deprecated URLField.verify_exists. [auginst]	
contrib	20 hours ago	Fixed #17864 – Added Hong Kong localflavor. Thanks to mrkschan and A... [julien]	
core	2 days ago	Fixed #18035 – Removed deprecated AdminMediaHandler, as per official... [claudep]	
db	2 days ago	Fixed #17877 – Ensured that extra WHERE clauses get correctly ANDed ... [julien]	
dispatch	9 months ago	Fixed #16225 – Removed unused imports. Many thanks to Aymeric August... [jezdez]	
forms	5 days ago	Removed pre-2.6 compatibility code in date-based form fields. Refs #9... [claudep]	
http	10 days ago	Removed deprecated CompatCookie. [auginst]	
middleware	a month ago	Fixed #17817 – Modified LocalMiddleware to use full URLs when redire... [jezdez]	
shortcuts	a year ago	Fixed #15010 – Added current_app parameter to close gap between Temp... [jezdez]	
template	3 days ago	Fixed #15683 – Prevented escaped string to be needlessly marked safe... [claudep]	
templatetags	11 days ago	Removed with_statement imports, useless in Python >= 2.6. Refs #17965... [claudep]	
test	a day ago	Fixed #17848 – Added setting_changed signal for cases when TEMPLATE_... [claudep]	

It's 07:47:41 PM



- home
- [dinosaurs](#)
- [aliens](#)

pjax loads html from your server into the current page without a full page load. It's ajax with real permalinks, page titles, and a working back button that fully degrades.

Check the box to toggle pjax.

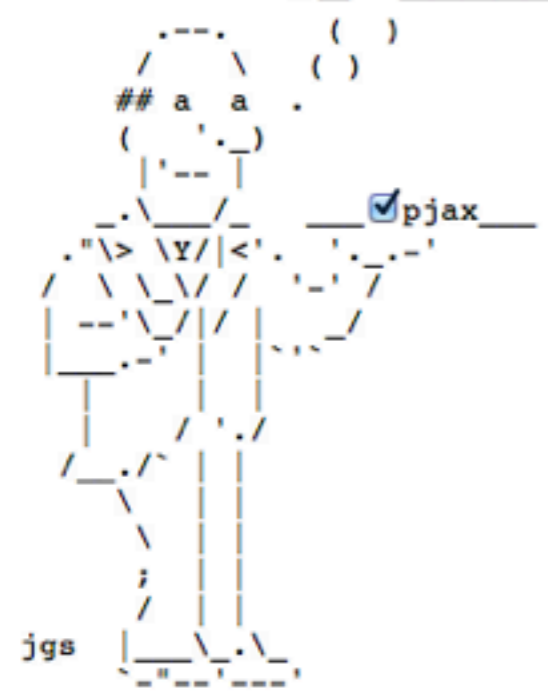
Whenever the time changes, a full page load has happened. If the time doesn't change, no full page load has occurred.

github.com/defunkt/jquery-pjax

The idea is you can't tell the difference between pjax page loads and normal page loads. On complicated sites, browsing just "feels faster."

[view this example's source code](#)

It's 07:47:41 PM

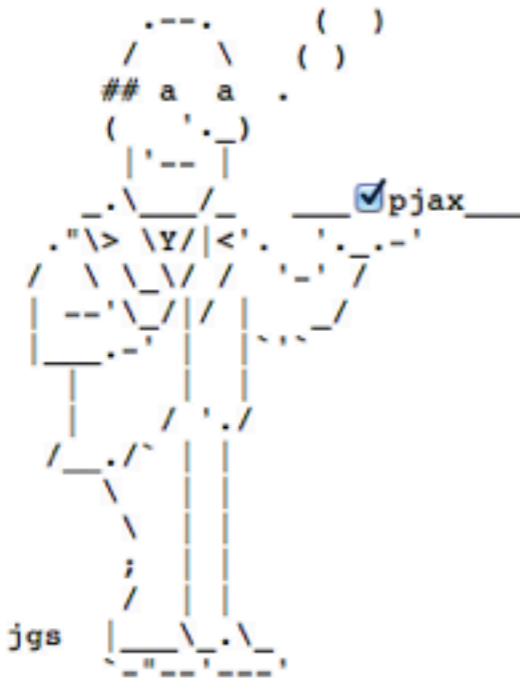


- [home](#)
- [dinosaurs](#)
- [aliens](#)

github.com/defunkt/jquery-pjax



It's 07:47:41 PM



- [home](#)
- [dinosaurs](#)
- [aliens](#)

github.com/defunkt/jquery-pjax



defunkt/jquery-pjax

GitHub, Inc. [US] <https://github.com/defunkt/jquery-pjax>

github Search... Explore Gist Blog Help adrianholovaty 14

defunkt / jquery-pjax Watch Fork 3,748 184

Code Network Pull Requests 5 Issues 11 Stats & Graphs

pushState + ajax = pjax — [Read more](#)
<http://pjax.herokuapp.com>

Clone in Mac ZIP HTTP Git Read-Only Read-Only access

branch: master Files Commits Branches 4 Tags Downloads

Latest commit to the master branch

Merge pull request #112 from hazzik/patch-1

josh authored 6 days ago commit c040777774

jquery-pjax /

name	age	message	history
test	6 days ago	Add X-PJAX-Container header [josh]	
.travis.yml	2 months ago	Silence server logs [josh]	
LICENSE	a year ago	pjax [defunkt]	
README.md	a month ago	Update README.md [oscardelben]	
jquery.pjax.js	6 days ago	Merge pull request #112 from hazzik/patch-1 [josh]	

One. Functionally obtrusive, loading the href with ajax into data-pjax:

```
<a href='/explore' data-pjax='#main'>Explore</a>
```

```
$( 'a[data-pjax]' ).pjax()
```

<https://github.com/defunkt/jquery-pjax>

Two. Slightly obtrusive, passing a container and binding an error handler:

```
<a href='/explore' class='js-pjax'>Explore</a>
```

```
$('.js-pjax').pjax('#main')

$('#main').live('pjax:error', function(e, xhr, err) {
  $('#error').text('Something went wrong: ' + err)
})
```

<https://github.com/defunkt/jquery-pjax>

Three. Unobtrusive, showing a 'loading' spinner:

```
<div id='main'>
  <div class='loader' style='display:none'><img src='spin.gif'></div>
  <div class='tabs'>
    <a href='/explore'>Explore</a>
    <a href='/help'>Help</a>
  </div>
</div>
```

```
$( 'a' ).pjax( '#main' ).live( 'click', function() {
  $( this ).showLoader()
})
```

<https://github.com/defunkt/jquery-pjax>

```
def my_view(request):
    if request.META.get('HTTP_X_PJAX'):
        template = 'content-pjax.html'
    else:
        template = 'content.html'

    # Get data...

    return render_to_response(template, context)
```

github [Explore](#) [Gist](#) [Blog](#) [Help](#) adrianholovaty


[adrianholovaty / django-pancake](#) [Admin](#) [Unwatch](#) [Fork](#) [Pull Request](#) 27 4

Code Network Pull Requests 1 Issues 1 Wiki 0 Stats & Graphs

branch: master Files Commits Branches 3 Tags Downloads

django-pancake / django_pancake

name	age	message	history
..			
__init__.py	2 months ago	Added stub flatten() function and stub tests [adrianholovaty]	
__init__.pyc	2 months ago	Added stub flatten() function and stub tests [adrianholovaty]	
flatten.py	2 months ago	Fixed bug where (% load %) tags were not carried up from (% include %... [adrianholovaty]	
make_pancakes.py	2 months ago	Added make_pancakes.py [adrianholovaty]	



GitHub

- [About](#)
- [Blog](#)
- [Features](#)
- [Contact & Support](#)
- [Training](#)
- [GitHub Enterprise](#)
- [Site Status](#)

Tools


- [Gauges: Analyze web traffic](#)
- [Speaker Deck: Presentations](#)
- [Gist: Code snippets](#)
- [GitHub for Mac](#)
- [Issues for iPhone](#)
- [Job Board](#)

Extras

- [GitHub Shop](#)
- [The Octodex](#)

Documentation

- [GitHub Help](#)
- [Developer API](#)
- [GitHub Flavored Markdown](#)
- [GitHub Pages](#)

 [Terms of Service](#) [Privacy](#) [Security](#)
© 2012 GitHub Inc. All rights reserved.

VS.

name	age	message	history
..			
__init__.py	2 months ago	Added stub flatten() function and stub tests [adrianholovaty]	
__init__.pyc	2 months ago	Added stub flatten() function and stub tests [adrianholovaty]	
flatten.py	2 months ago	Fixed bug where (% load %) tags were not carried up from (% include %... [adrianholovaty]	
make_pancakes.py	2 months ago	Added make_pancakes.py [adrianholovaty]	


```
from djpjax import pjax

@pjax()
def my_view(request):
    # Get data...
    template = 'content.html'
    return render_to_response(template, context)
```

<https://github.com/jacobian/django-pjax>

Problem #1: Multiple content boxes

Log in / create account

Article Talk Read Edit View history Search

Web page

From Wikipedia, the free encyclopedia

A **web page** or **webpage** is a document or information resource that is suitable for the World Wide Web and can be accessed through a web browser and displayed on a monitor or mobile device. This information is usually in HTML or XHTML format, and may provide navigation to other web pages via hypertext links. Web pages frequently subsume other resources such as style sheets, scripts and images into their final presentation.

Web pages may be retrieved from a local computer or from a remote web server. The web server may restrict access only to a private network, e.g. a corporate intranet, or it may publish pages on the World Wide Web. Web pages are requested and served from web servers using Hypertext Transfer Protocol (HTTP).

Web pages may consist of files of static text and other content stored within the web server's file system (static web pages), or may be constructed by server-side software when they are requested (dynamic web pages). Client-side scripting can make web pages more responsive to user input once on the client browser.

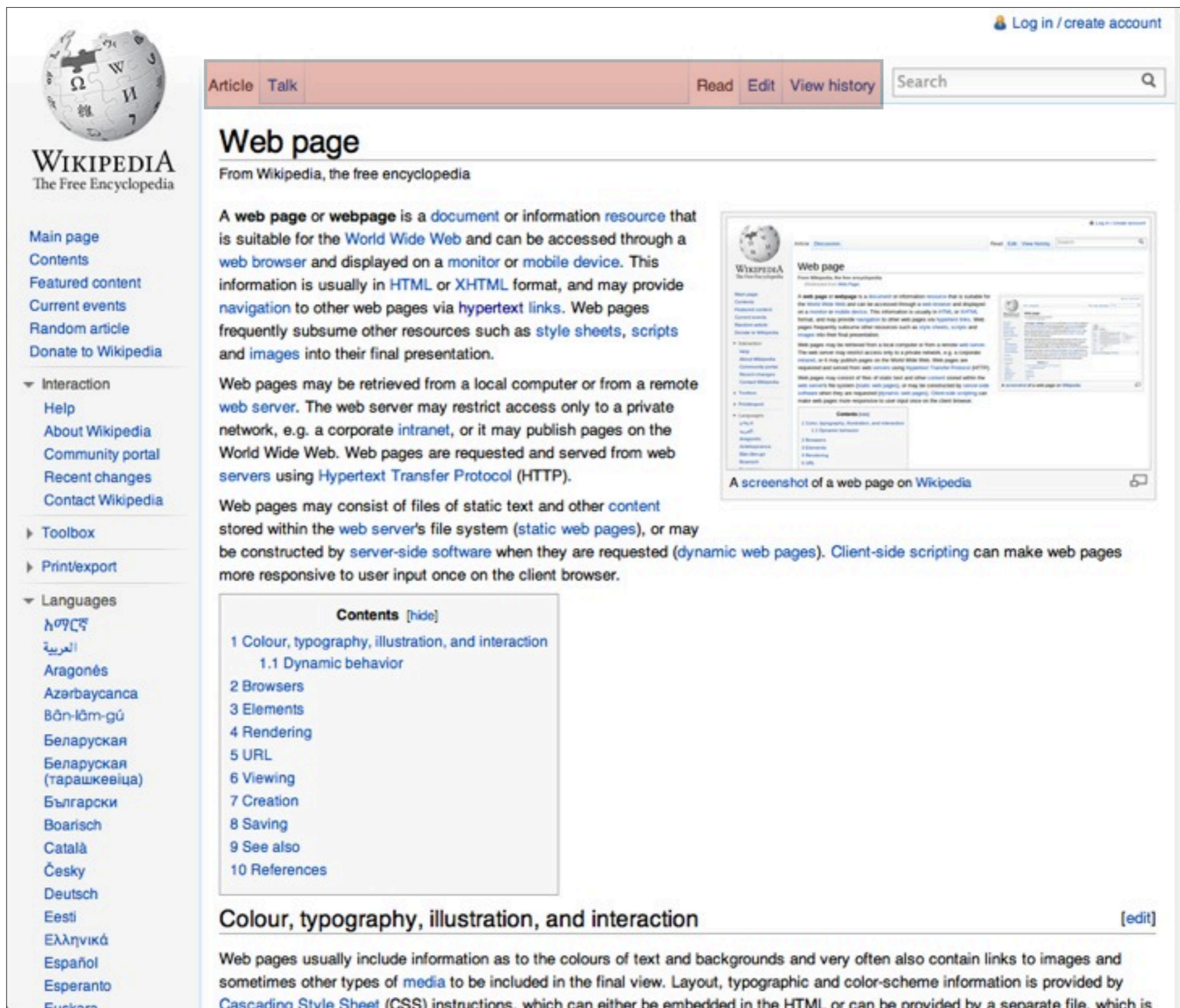
Contents [hide]

- Colour, typography, illustration, and interaction
 - Dynamic behavior
- Browsers
- Elements
- Rendering
- URL
- Viewing
- Creation
- Saving
- See also
- References

Colour, typography, illustration, and interaction [edit]

Web pages usually include information as to the colours of text and backgrounds and very often also contain links to images and sometimes other types of media to be included in the final view. Layout, typographic and color-scheme information is provided by Cascading Style Sheet (CSS) instructions, which can either be embedded in the HTML or can be provided by a separate file, which is

Problem #2: Non-visible changes



The image shows a screenshot of the Wikipedia article titled "Web page". The page layout includes a top navigation bar with "Article" and "Talk" tabs, and "Read", "Edit", and "View history" buttons. A search box is located on the right. The main content area starts with the title "Web page" and a sub-header "From Wikipedia, the free encyclopedia". The text defines a web page as a document or information resource suitable for the World Wide Web, accessed via a web browser. It mentions that web pages are usually in HTML or XHTML format and can include navigation links, style sheets, scripts, and images. A paragraph explains that web pages can be retrieved from a local computer or a remote web server, and that servers may restrict access to private networks or publish pages on the World Wide Web. Another paragraph states that web pages can be static or dynamic, and that client-side scripting can make them more responsive. A table of contents is provided, listing sections from "Colour, typography, illustration, and interaction" to "References". A screenshot of the article is shown in a smaller window, with the caption "A screenshot of a web page on Wikipedia".

Log in / create account

Article Talk Read Edit View history Search

Web page

From Wikipedia, the free encyclopedia

A **web page** or **webpage** is a **document** or information **resource** that is suitable for the **World Wide Web** and can be accessed through a **web browser** and displayed on a **monitor** or **mobile device**. This information is usually in **HTML** or **XHTML** format, and may provide **navigation** to other web pages via **hypertext links**. Web pages frequently subsume other resources such as **style sheets**, **scripts** and **images** into their final presentation.

Web pages may be retrieved from a local computer or from a remote **web server**. The web server may restrict access only to a private network, e.g. a corporate **intranet**, or it may publish pages on the World Wide Web. Web pages are requested and served from web servers using **Hypertext Transfer Protocol (HTTP)**.

Web pages may consist of files of static text and other **content** stored within the **web server's** file system (**static web pages**), or may be constructed by **server-side software** when they are requested (**dynamic web pages**). **Client-side scripting** can make web pages more responsive to user input once on the client browser.

Contents [hide]

- 1 Colour, typography, illustration, and interaction
 - 1.1 Dynamic behavior
- 2 Browsers
- 3 Elements
- 4 Rendering
- 5 URL
- 6 Viewing
- 7 Creation
- 8 Saving
- 9 See also
- 10 References

Colour, typography, illustration, and interaction [edit]

Web pages usually include information as to the colours of text and backgrounds and very often also contain links to images and sometimes other types of **media** to be included in the final view. Layout, typographic and color-scheme information is provided by **Cascading Style Sheet (CSS)** instructions, which can either be embedded in the HTML or can be provided by a separate file, which is

Problem #3: Context-switching

Python ●

JavaScript

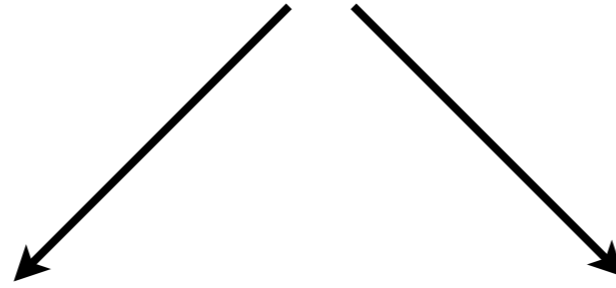
SQL



Problem #4: Computers should do this



Two pages, one system



Web page - Wikipedia, the free encyclopedia

en.wikipedia.org/wiki/Web_page

Log in / create account

Article Talk

Read Edit View history Search

Web page

From Wikipedia, the free encyclopedia

A **web page** or **webpage** is a document or information resource that is suitable for the World Wide Web and can be accessed through a web browser and displayed on a monitor or mobile device. This information is usually in HTML or XHTML format, and may provide navigation to other web pages via [hypertext](#) links. Web pages frequently subsume other resources such as [style sheets](#), [scripts](#) and [images](#) into their final presentation.

Web pages may be retrieved from a local computer or from a remote [web server](#). The web server may restrict access only to a private network, e.g. a corporate [intranet](#), or it may publish pages on the World Wide Web. Web pages are requested and served from web servers using [Hypertext Transfer Protocol \(HTTP\)](#).

Web pages may consist of files of static text and other content stored within the web server's file system ([static web pages](#)), or may be constructed by [server-side software](#) when they are requested ([dynamic web pages](#)). [Client-side scripting](#) can make web pages more responsive to user input once on the client browser.

Contents [hide]

- 1 Colour, typography, illustration, and interaction
 - 1.1 Dynamic behavior
- 2 Browsers
- 3 Elements
- 4 Rendering
- 5 URL
- 6 Viewing
- 7 Creation
- 8 Saving
- 9 See also
- 10 References

Colour, typography, illustration, and interaction

[edit]

Web pages usually include information as to the colours of text and backgrounds and very often also contain links to images and sometimes other types of media. As an included in the final view, layout, typographic and color-scheme information is provided by [Cascading Style Sheet \(CSS\)](#) instructions, which

Hypertext - Wikipedia, the free encyclopedia

en.wikipedia.org/wiki/Hypertext

Log in / create account

Article Talk

Read Edit View history Search

Hypertext

From Wikipedia, the free encyclopedia

"[Metatext](#)" redirects here. For the literary concept, see [Metafiction](#).

Hypertext is text displayed on a computer or other electronic device with references ([hyperlinks](#)) to other text that the reader can immediately access, usually by a mouse click or keypress sequence. Apart from running text, hypertext may contain tables, images and other presentational devices. Hypertext is the underlying concept defining the structure of the [World Wide Web](#).^[1] It is an easy-to-use and flexible format to share information over the [Internet](#).

Contents [hide]

- 1 Etymology
- 2 Types and uses of hypertext
- 3 History
- 4 Implementations
- 5 Academic conferences
- 6 Hypertext fiction
 - 6.1 Critics and theorists
- 7 See also
- 8 References
- 9 Further reading
- 10 External links

Etymology

[edit]

The prefix **hyper-** (comes from the Greek prefix "υπερ-" and means "over" or "beyond", while having common origins with the English word "super") signifies the overcoming of the old linear constraints of written text. The term "hypertext" is often used where the term "[hypermedia](#)" might seem appropriate. In 1962, author [Ted Nelson](#) – who coined both terms in 1963 – wrote:

By now the word "hypertext" has become generally accepted for branching and responding text, but the corresponding word "hypermedia", meaning complexes of branching and responding graphics, movies and sound – as well as text – is much less used. Instead they use the strange term "interactive multimedia": this is four syllables longer, and does not express the idea of extending hypertext.

– Nelson, *Literary Machines*, 1992

Types and uses of hypertext

[edit]

Hypertext documents can either be static (prepared and stored in advance) or dynamic (continually changing in response to user input). Static hypertext can

The Solution

Auto-pjax
tinyurl.com/django-pjax



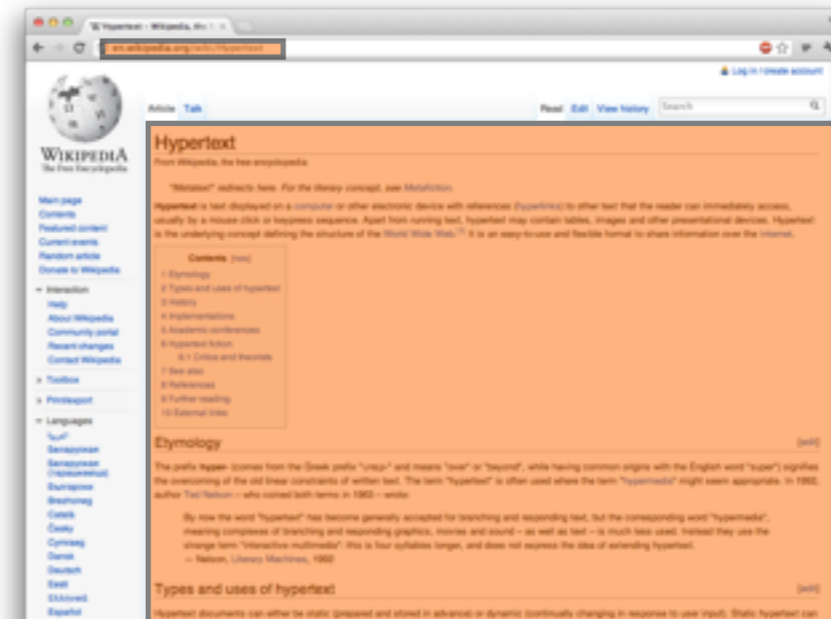
Goal #1: Auto diffs via Ajax



Web framework
returns full page



Web framework
returns *diff*



Goal #2: Support multiple content boxes

The image shows a screenshot of a Wikipedia article titled "Web page". The page layout includes a left sidebar with navigation links, a main content area with text and a table of contents, and a right sidebar with a search bar and user options. The article text defines a web page as a document accessible via a web browser, typically in HTML or XHTML format. It also discusses how web pages are retrieved from servers and how they can be static or dynamic. A table of contents is provided, listing sections from "Colour, typography, illustration, and interaction" to "References". A screenshot of the article itself is embedded within the text.

WIKIPEDIA
The Free Encyclopedia

Log in / create account

Article Talk

Read Edit View history Search

Web page

From Wikipedia, the free encyclopedia

A **web page** or **webpage** is a document or information resource that is suitable for the World Wide Web and can be accessed through a web browser and displayed on a monitor or mobile device. This information is usually in HTML or XHTML format, and may provide navigation to other web pages via **hypertext links**. Web pages frequently subsume other resources such as **style sheets**, **scripts** and **images** into their final presentation.

Web pages may be retrieved from a local computer or from a remote web server. The web server may restrict access only to a private network, e.g. a corporate intranet, or it may publish pages on the World Wide Web. Web pages are requested and served from web servers using **Hypertext Transfer Protocol (HTTP)**.

Web pages may consist of files of static text and other content stored within the web server's file system (**static web pages**), or may be constructed by **server-side software** when they are requested (**dynamic web pages**). Client-side scripting can make web pages more responsive to user input once on the client browser.

Contents [hide]

- Colour, typography, illustration, and interaction
 - Dynamic behavior
- Browsers
- Elements
- Rendering
- URL
- Viewing
- Creation
- Saving
- See also
- References

Colour, typography, illustration, and interaction [edit]

Web pages usually include information as to the colours of text and backgrounds and very often also contain links to images and sometimes other types of **media** to be included in the final view. Layout, typographic and color-scheme information is provided by **Cascading Style Sheet (CSS)** instructions, which can either be embedded in the HTML or can be provided by a separate file, which is

Goal #3: Support non-visible changes (JS!)

The image shows a screenshot of the Wikipedia article titled "Web page". The page layout includes a top navigation bar with "Article" and "Talk" tabs, and "Read", "Edit", and "View history" buttons. A search box is located on the right. The main content area starts with the title "Web page" and a sub-header "From Wikipedia, the free encyclopedia". The text defines a web page as a document or information resource suitable for the World Wide Web, accessed via a web browser. It mentions that web pages are usually in HTML or XHTML format and can include navigation links, style sheets, scripts, and images. A paragraph explains that web pages can be retrieved from a local computer or a remote web server, and that they are requested and served using HTTP. Another paragraph states that web pages can be static or dynamic, and that client-side scripting can make them more responsive. A table of contents is provided, listing sections from "1 Colour, typography, illustration, and interaction" to "10 References". A section titled "Colour, typography, illustration, and interaction" is expanded, showing text about CSS instructions. A sidebar on the left contains navigation links like "Main page", "Contents", and "Featured content". A language selection menu is at the bottom left. A "Log in / create account" link is in the top right. A small inset image shows a screenshot of the same article on a mobile device.

Log in / create account

Article Talk Read Edit View history Search

Web page

From Wikipedia, the free encyclopedia

A **web page** or **webpage** is a **document** or information **resource** that is suitable for the **World Wide Web** and can be accessed through a **web browser** and displayed on a **monitor** or **mobile device**. This information is usually in **HTML** or **XHTML** format, and may provide **navigation** to other web pages via **hypertext links**. Web pages frequently subsume other resources such as **style sheets**, **scripts** and **images** into their final presentation.

Web pages may be retrieved from a local computer or from a remote **web server**. The web server may restrict access only to a private network, e.g. a corporate **intranet**, or it may publish pages on the World Wide Web. Web pages are requested and served from web servers using **Hypertext Transfer Protocol (HTTP)**.

Web pages may consist of files of static text and other **content** stored within the **web server's** file system (**static web pages**), or may be constructed by **server-side software** when they are requested (**dynamic web pages**). **Client-side scripting** can make web pages more responsive to user input once on the client browser.

Contents [hide]
1 Colour, typography, illustration, and interaction
1.1 Dynamic behavior
2 Browsers
3 Elements
4 Rendering
5 URL
6 Viewing
7 Creation
8 Saving
9 See also
10 References

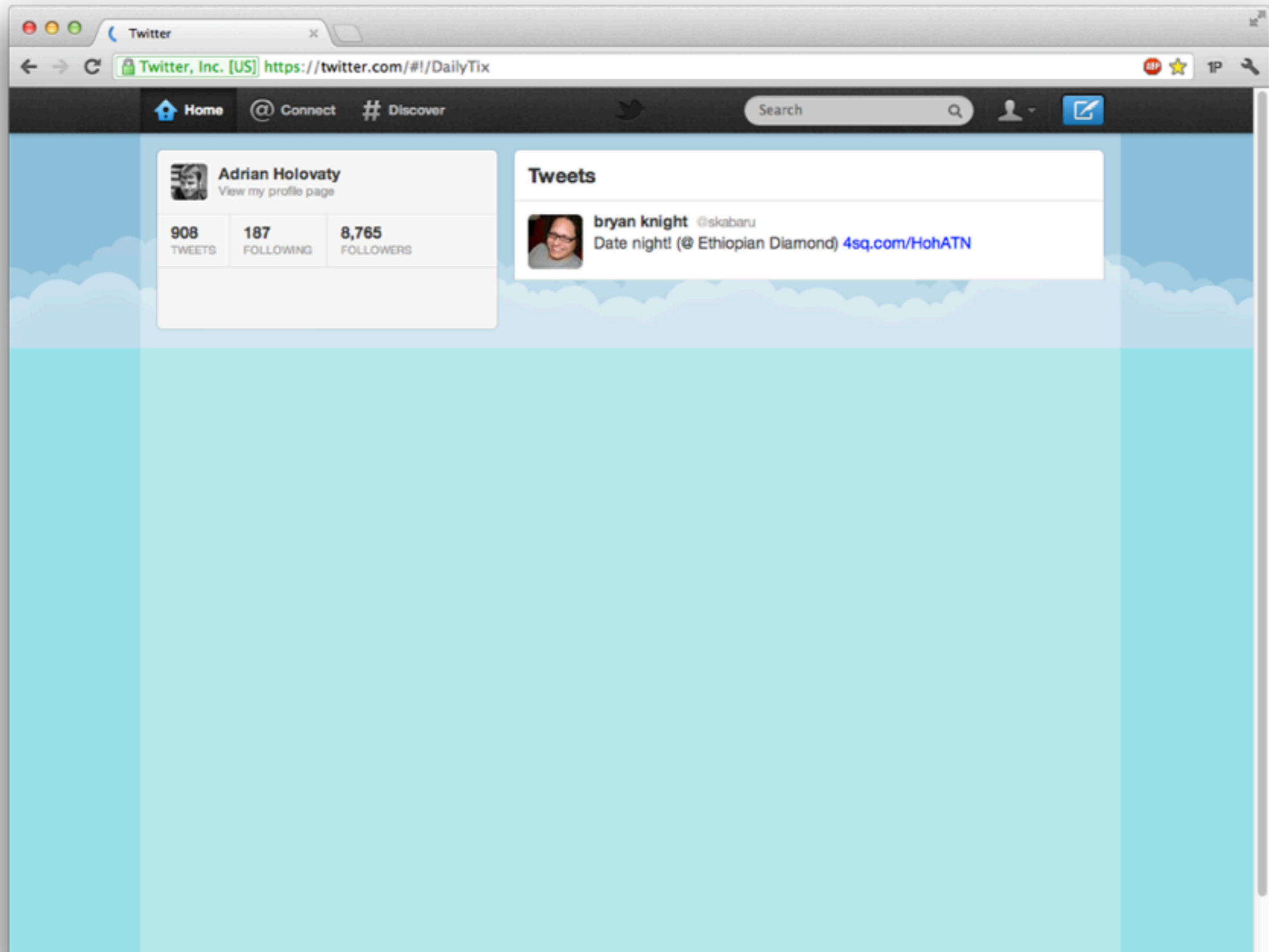
Colour, typography, illustration, and interaction [edit]

Web pages usually include information as to the colours of text and backgrounds and very often also contain links to images and sometimes other types of **media** to be included in the final view. Layout, typographic and color-scheme information is provided by **Cascading Style Sheet (CSS)** instructions, which can either be embedded in the HTML or can be provided by a separate file, which is

Goal #4: Respect URLs and permalinks

~~twitter.com/#!/adrianholovaty/status/18947108~~

Goal #5: Don't require JS on initial page load



Goal #6: Bail if needed

The screenshot shows the homepage of EveryBlock Chicago. At the top, there is a search bar with the text "e.g., 1060 W. Addison, 60615, Humboldt Park" and a "Search" button. Below the search bar is a navigation menu with "Log in" on the right. The main content area is divided into four columns, each with a heading and a brief description:

- Follow your favorite places:** Pick a neighborhood, block or ZIP — or create a personalized area. Sign up for one or many.
- Learn what's happening:** Read nearby news from hundreds of sources. Get updates via e-mail or your custom homepage.
- Share with neighbors:** Start a discussion, share an announcement, ask your neighbors a question, or answer one of theirs.
- Your block gets better:** Exchange ideas. Gain recognition. Solve problems. Be a better neighbor.

Below these columns is a "Sign up for free" section with a "Go" button and options to "Sign in with Facebook" or "Sign in with Twitter". There is also a "Like" button and a note "5020 likes. Sign Up to see what your friends like." On the left side, there is a "Stories popular with our users" section with a "View more" link. The stories listed are:

- Gay/Lesbian Friendly Churches in Rogers Park/Edgewater:** Does anyone know of any Gay/Lesbian Friendly Churches in Rogers Park or Edgewater? I recently found out the Rogers ...
- Moving to Andersonville area!** Hi All, I'm so glad I found this web resource! My husband and I will be moving to Chicago this ...
- Homeless Lady:** Does anyone have information on the presumably homeless lady in/around Roscoe Village who can be found around

The screenshot shows a specific announcement on the EveryBlock Chicago website. The URL in the browser is "chicago.everyblock.com/announcements/apr09-moving-andersonville-area-4889923/". The announcement is titled "Moving to Andersonville area!" and is posted by Kristin. The main text of the announcement reads:

Hi All,
I'm so glad I found this web resource!
My husband and I will be moving to Chicago this summer, and several friends have recommended Rogers Park and Andersonville. Any great apartment-finder resources or realtors you can recommend? Any areas to avoid? I'll be a prof at Loyola, so I'd like an area that is more family-focused and liberal. Thanks so much, and looking forward to joining you all!

Below the announcement is a "25 COMMENTS" section. The first comment is from Ann Hinterman, who says: "Welcome to the hood, Kristin! If you're looking at Rogers Park (and you should!), the Rogers Park Builders Group is a local association of responsible building owners/landlords and realtors. I'm sure someone there could help you out!" The second comment is from Rebecca Kell, who says: "Andersonville is family-friendly, but more expensive than Loyola. It's also not on the CTA trains. It has buses, though. Both neighborhoods seem fairly liberal. Andersonville is the 2nd big LGBT neighborhood in the city and Rogers Park is being considered the new, 3rd LGBT neighborhoods."

On the right side of the announcement, there is a "SUBSCRIBE" button and a note "12 neighbors are subscribed to this conversation." At the bottom right, there is a "This was posted to Rogers Park" notification.

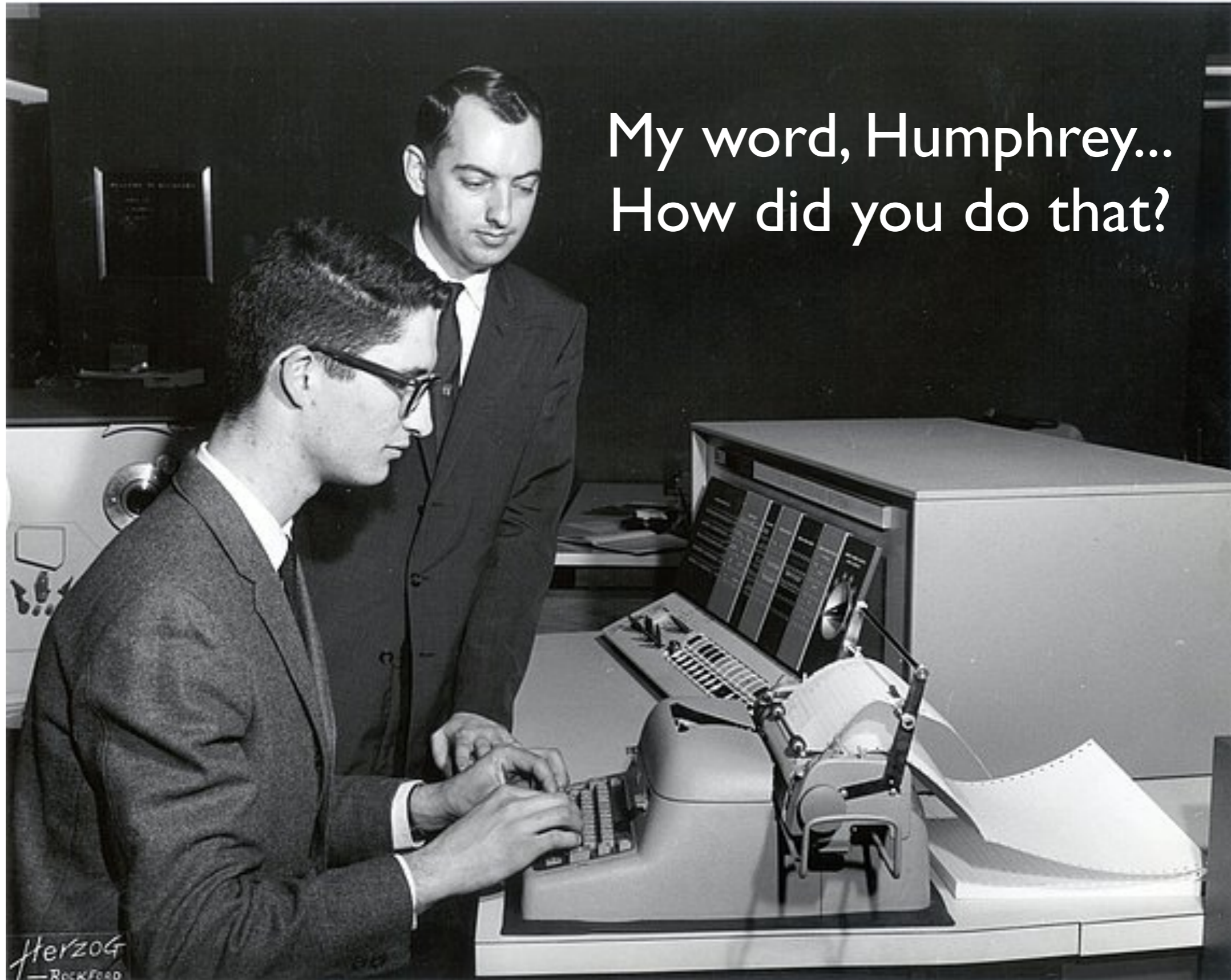
Goal #7: Security

Any application-specific data
that's revealed should be
safe for public consumption.

Goal #8: Favor correctness, win devs' trust



Goal #9: Amaze



My word, Humphrey...
How did you do that?

Why pass HTML instead of structured data?

Structured data

```
{"title": "Web page",  
"content": "A web page or webpage is a document or...",  
"last_edited": "2012-03-25"}
```

- Then you need client-side templating.
- Optimize for server-side development, because you have to do it anyway. And it's more fun. :-)

Example



The image shows a screenshot of a Wikipedia article titled "Web page". The page layout includes a sidebar on the left with the Wikipedia logo and navigation links, a main content area with a title and introductory text, and a table of contents. A screenshot of the article is also embedded within the main content area.

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Toolbox

Print/export

Languages
አማርኛ
العربية
Aragonés
Azərbaycanca
Бân-lâm-gú
Беларуская
Беларуская (тарашкевіца)
Български
Boarisch
Català
Česky
Deutsch
Eesti
Ελληνικά
Español
Esperanto
Euskara

Article **Talk** Read Edit View history Search

Web page

From Wikipedia, the free encyclopedia

A **web page** or **webpage** is a **document** or information **resource** that is suitable for the **World Wide Web** and can be accessed through a **web browser** and displayed on a **monitor** or **mobile device**. This information is usually in **HTML** or **XHTML** format, and may provide **navigation** to other web pages via **hypertext links**. Web pages frequently subsume other resources such as **style sheets**, **scripts** and **images** into their final presentation.

Web pages may be retrieved from a local computer or from a remote **web server**. The web server may restrict access only to a private network, e.g. a corporate **intranet**, or it may publish pages on the World Wide Web. Web pages are requested and served from web servers using **Hypertext Transfer Protocol (HTTP)**.


Web pages may consist of files of static text and other **content** stored within the **web server's** file system (**static web pages**), or may be constructed by **server-side software** when they are requested (**dynamic web pages**). **Client-side scripting** can make web pages more responsive to user input once on the client browser.

Contents [hide]

- 1 Colour, typography, illustration, and interaction
 - 1.1 Dynamic behavior
- 2 Browsers
- 3 Elements
- 4 Rendering
- 5 URL
- 6 Viewing
- 7 Creation
- 8 Saving
- 9 See also
- 10 References

Colour, typography, illustration, and interaction [edit]

Web pages usually include information as to the colours of text and backgrounds and very often also contain links to images and sometimes other types of **media** to be included in the final view. Layout, typographic and color-scheme information is provided by **Cascading Style Sheet (CSS)** instructions, which can either be embedded in the HTML or can be provided by a separate file, which is



A screenshot of a web page on Wikipedia

I. Find all on-site links

The image shows a screenshot of the Wikipedia article titled "Web page". The page layout includes a top navigation bar with "Article", "Talk", "Read", "Edit", "View history", and a search box. The main content area contains several paragraphs of text, a table of contents, and a section header. A sidebar on the left contains navigation links, and a bottom right corner has an "[edit]" button. A screenshot of the article is also shown within the main content area.

Log in / create account

Article **Talk** Read **Edit** **View history** Search

Web page

From Wikipedia, the free encyclopedia

A **web page** or **webpage** is a **document** or information **resource** that is suitable for the **World Wide Web** and can be accessed through a **web browser** and displayed on a **monitor** or **mobile device**. This information is usually in **HTML** or **XHTML** format, and may provide **navigation** to other web pages via **hypertext links**. Web pages frequently subsume other resources such as **style sheets**, **scripts** and **images** into their final presentation.

Web pages may be retrieved from a local computer or from a remote **web server**. The web server may restrict access only to a private network, e.g. a corporate **intranet**, or it may publish pages on the World Wide Web. Web pages are requested and served from web servers using **Hypertext Transfer Protocol (HTTP)**.

Web pages may consist of files of static text and other **content** stored within the **web server's** file system (**static web pages**), or may be constructed by **server-side software** when they are requested (**dynamic web pages**). **Client-side scripting** can make web pages more responsive to user input once on the client browser.

Contents [hide]

- 1 Colour, typography, illustration, and interaction
 - 1.1 Dynamic behavior
- 2 Browsers
- 3 Elements
- 4 Rendering
- 5 URL
- 6 Viewing
- 7 Creation
- 8 Saving
- 9 See also
- 10 References

Colour, typography, illustration, and interaction

Web pages usually include information as to the colours of text and backgrounds and very often also contain links to images and sometimes other types of **media** to be included in the final view. Layout, typographic and color-scheme information is provided by **Cascading Style Sheet (CSS)** instructions, which can either be embedded in the HTML or can be provided by a separate file, which is

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Toolbox
Print/export

Languages
አማርኛ
العربية
Aragonés
Azərbaycanca
Bân-lâm-gú
Беларуская
Беларуская (тарашкевіца)
Български
Boarisch
Català
Česky
Deutsch
Eesti
Ελληνικά
Español
Esperanto
Euskara

A screenshot of a web page on Wikipedia

[edit]

2. Set onclick

Load URL with Ajax:

- Pass X-PJAX header

- Pass current URL

When response comes in:

- Replace diffs (**MAGIC!**)

On error:

- Load next page manually

3. Django middleware

If request has X-PJAX header:
Calculate diff (**MAGIC!**)
Return JSON of diff

Overview



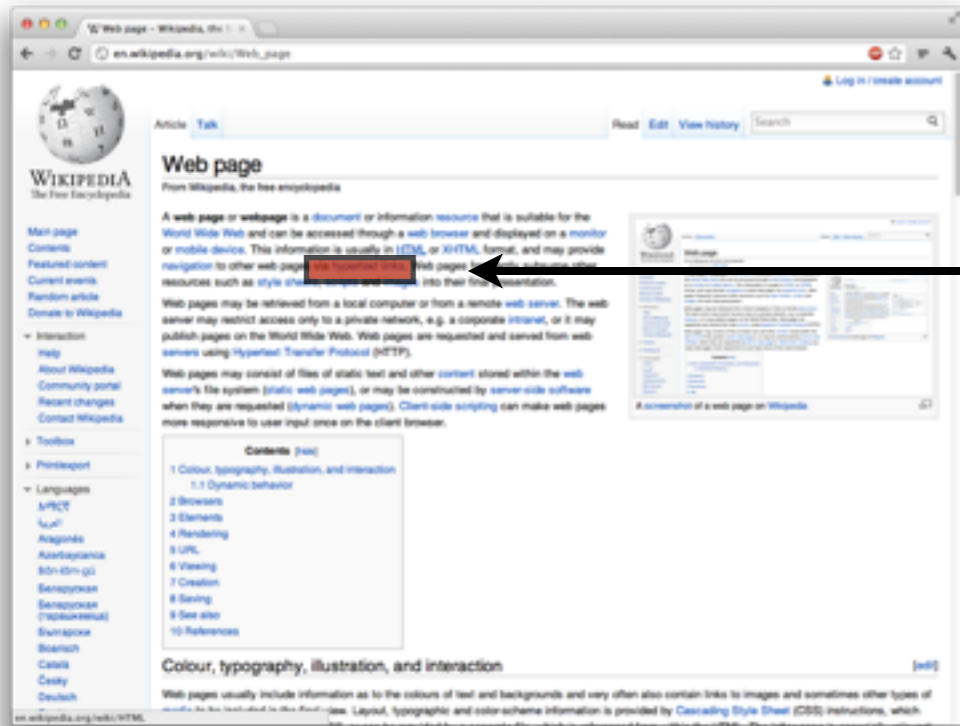
Overview



Click link

Trigger Ajax request

Overview



Click link

Trigger Ajax request



Server says: "These bits changed"

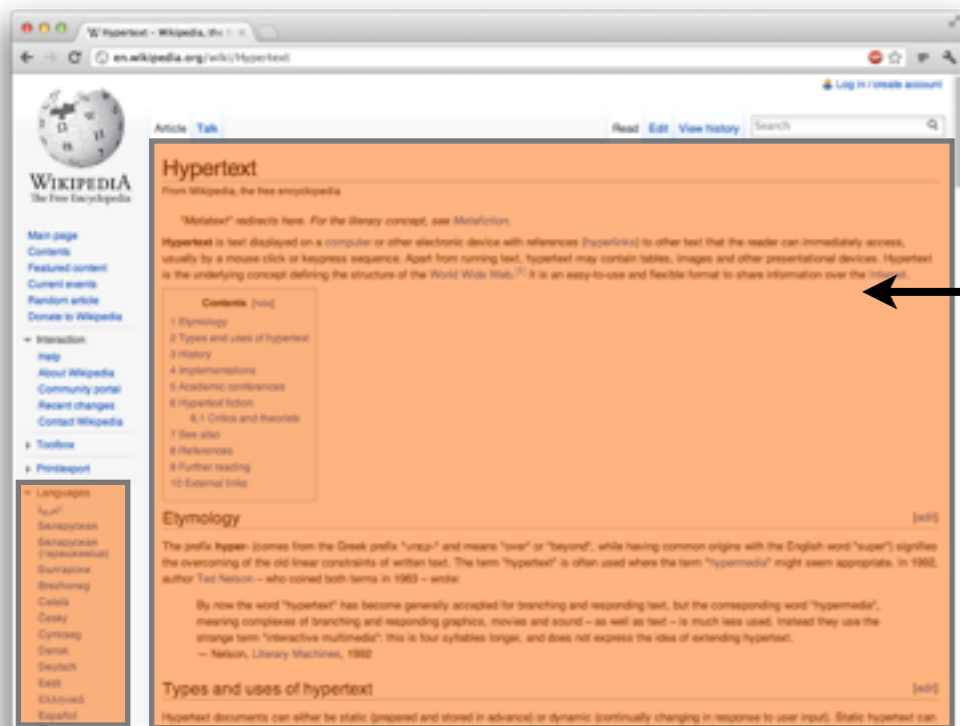
Overview



Click link

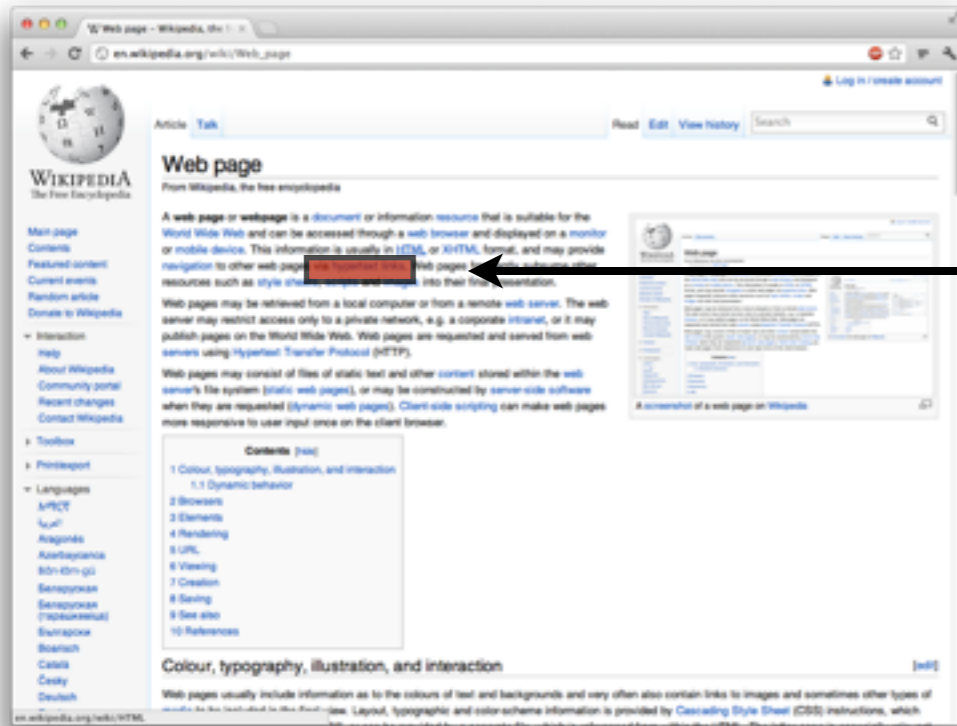
Trigger Ajax request

Server says: "These bits changed"



JavaScript changes the page in place

Overview

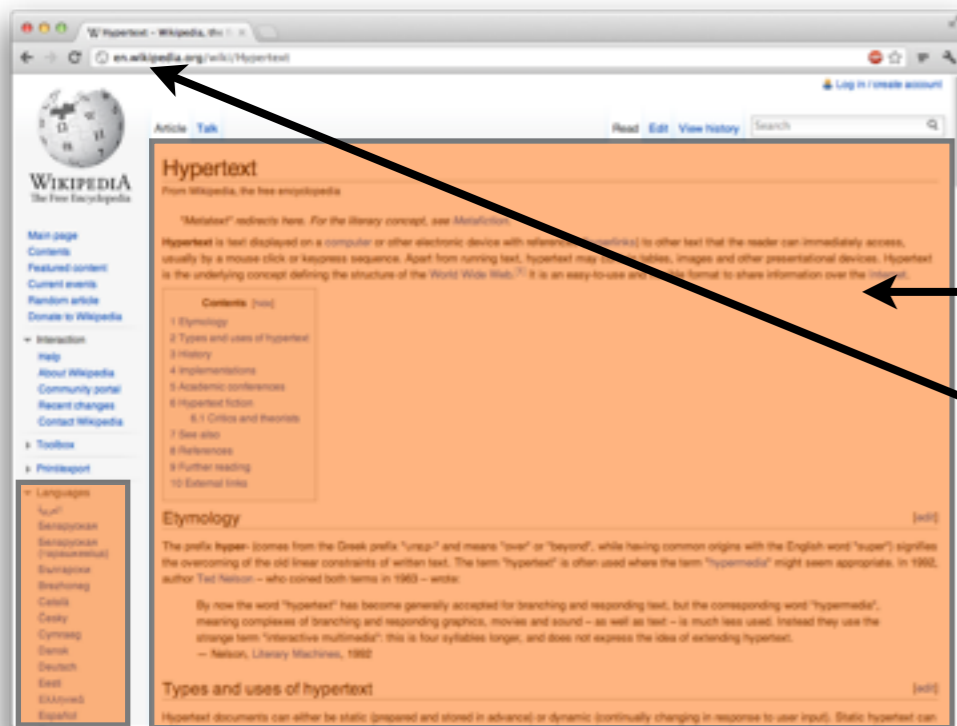


Click link

Trigger Ajax request



Server says: "These bits changed"



JavaScript changes the page in place

URL gets updated with pushstate

Two bits of MAGIC

- Server: Which bits are different?
- Client: How do we update the HTML document?

Django template inheritance

base.html

```
<html>
<body>
<h1>{% block h1 %}{% endblock %}</h1>

<div id="content">
{% block content %}{% endblock %}
</div>
```

Django template inheritance

base.html

```
<html>
<body>
<h1>{% block h1 %}{% endblock %}</h1>

<div id="content">
{% block content %}{% endblock %}
</div>
```

child1.html

```
{% extends "base.html" %}

{% block h1 %}
First child
{% endblock %}

{% block content %}
<p>Welcome to the first child!</p>
{% endblock %}
```

Django template inheritance

base.html

```
<html>
<body>
<h1>{% block h1 %}{% endblock %}</h1>

<div id="content">
{% block content %}{% endblock %}
</div>
```

child1.html

```
{% extends "base.html" %}

{% block h1 %}
First child
{% endblock %}

{% block content %}
<p>Welcome to the first child!</p>
{% endblock %}
```

child2.html

```
{% extends "base.html" %}

{% block h1 %}
Second child
{% endblock %}

{% block content %}
<p>Welcome to the second kid!</p>
{% endblock %}
```


Django template inheritance

base.html

```
<html><body><h1> h1 </h1><div id="content"> content </div>
```

child1.html

```
h1 First child
```

```
content <p>Welcome to the first child!</p>
```

child2.html

```
h1 Second child
```

```
content <p>Welcome to the second kid!</p>
```

Basic diff using template inheritance

<http://example.com/child1/>

```
<html><body><h1>
```

```
h1 First child
```

```
</h1><div id="content">
```

```
content <p>Welcome to the first child!</p>
```

```
</div>
```

<http://example.com/child2/>

```
{  
  "block:h1": "First child",  
  "block:content": "<p>Welcome to the second kid!</p>"  
}
```

Hooking into the HTML: Option 1

<http://example.com/child1/>

```
<html><body><h1>
```

```
<div id="dj_block_h1">
```

```
h1 First child
```

```
</div>
```

```
</h1><div id="content">
```

```
<div id="dj_block_content">
```

```
content <p>Welcome to the first child!</p>
```

```
</div>
```

```
</div>
```

Hooking into the HTML: Option 1

<http://example.com/child/>

```
<html><body><h1>
```

```
<div id="dj_block_h1">
```

```
h1 First child
```

```
</div>
```

```
</h1><div id="content">
```

```
<div id="dj_block_content">
```

```
content <p>Welcome to the first child!</p>
```

```
</div>
```

```
</div>
```

```
// data = from ajax.  
$('#dj_block_h1').html(data['block:h1']);  
$('#dj_block_content').html(data['block:content']);
```


Putting it all together

Template
(as written)

```
<html><body>  
<h1>{% block h1 %}{% endblock %}</h1>  
<div id="content">  
  {% block content %}{% endblock %}  
</div>
```

Putting it all together

Template
(as written)

```
<html><body>
<h1>{% block h1 %}{% endblock %}</h1>
<div id="content">
  {% block content %}{% endblock %}
</div>
```

Rendered
HTML

```
<html><body>
<h1><div id="dj_block_h1">First child</div></h1>
<div id="content"><div id="dj_block_content">
  <p>Welcome to the first child!</p>
</div></div>
```

h1

First child

content

<p>Welcome to the first child!</p>

Putting it all together

Template
(as written)

```
<html><body>
<h1>{% block h1 %}{% endblock %}</h1>
<div id="content">
  {% block content %}{% endblock %}
</div>
```

Rendered
HTML

```
<html><body>
<h1><div id="dj_block_h1">First child</div></h1>
<div id="content"><div id="dj_block_content">
  <p>Welcome to the first child!</p>
</div></div>
```

h1 First child **content** <p>Welcome to the first child!</p>

Ajax request

```
GET /page2/
X-pjax: /page1/
```

Putting it all together

Template
(as written)

```
<html><body>
<h1>{% block h1 %}{% endblock %}</h1>
<div id="content">
  {% block content %}{% endblock %}
</div>
```

Rendered
HTML

```
<html><body>
<h1><div id="dj_block_h1">First child</div></h1>
<div id="content"><div id="dj_block_content">
  <p>Welcome to the first child!</p>
</div></div>
```

h1	First child	content	<p>Welcome to the first child!</p>
----	-------------	---------	------------------------------------

Ajax request

```
GET /page2/
X-pjax: /page1/
```

Ajax response

```
{"dj_block_h1": "Second child",
"dj_block_content": "<p>Welcome to the second kid!</p>"}
```


Putting it all together

Template
(as written)

```
<html><body>
<h1>{% block h1 %}{% endblock %}</h1>
<div id="content">
  {% block content %}{% endblock %}
</div>
```

Rendered
HTML

```
<html><body>
<h1><div id="dj_block_h1">First child</div></h1>
<div id="content"><div id="dj_block_content">
  <p>Welcome to the first child!</p>
</div></div>
```

h1	First child	content	<p>Welcome to the first child!</p>
----	-------------	---------	------------------------------------

Ajax request

```
GET /page2/
X-pjax: /page1/
```

Ajax response

```
{"dj_block_h1": "Second child",
"dj_block_content": "<p>Welcome to the second kid!</p>"}
```

JS callback

```
$('#dj_block_h1').html(data['dj_block_h1']);
$('#dj_block_content').html(data['dj_block_content']);
history.pushState(null, document.title, "/page2/");
```

Template variables

```
def my_view(request):  
    return render_to_response('test.html', {  
        'name': request.GET.get('name'),  
    })
```

test.html

```
<html>  
<body>  
<h1>Hello, {{ name }}!</h1>
```

Template variables

```
<html>  
<body>  
<h1>Hello, {{ name }}!</h1>
```



```
<html><body><h1>Hello,
```

```
<span id="dj_var_name">
```

```
name
```

```
</span>
```

```
!</div>
```

```
// data = from ajax.  
$('#dj_var_name').html(data['dj_var_name']);
```

Then it starts to get tricky...

```
<html>
<head>
  <title>{{ title }}</title>
</head>
```

Can't put `` within `<title>`

Insertion JavaScript needs to special-case this:

```
document.title = 'New title';
```

How does the framework know this is a title??

How does the framework know this is a title?

```
<html>  
<head>  
  <title>{{ title title }}</title>  
</head>
```

Option 1: Provide hints in the template.

Requires extension of template syntax.

Requires manual work by template authors. :-)

How does the framework know this is a title?

```
<html>  
<head>  
  <title>{{ title }}</title>  
</head>
```

Option 2: Parse HTML.

Requires SUPER-MEGA-ALL-IN-ONE HTML,
JavaScript, CSS, Django template language
parser.

How does the framework know this is a title?

```
<html>
<head>
  <title>{{ title }}</title>
</head>
```

Option 3: **ALWAYS** change the `<title>`.

This is what `jquery-pjax` does.

Partial titles

```
<html>
<head>
  <title>{{ title }} | EveryBlock.com</title>
</head>
```

Can't put `` within `<title>`
document.title call now needs extra information

```
document.title = 'New title' + ' | EveryBlock.com ';
```

A good argument for always changing the `<title>`.

Snippets in HTML tags

```
<body class="{{ body_class }}">
  <p {{ p_attrs }}
    {{ content }}
  </p>
</body>
```

Can't put `` within attribute

How does the framework know these are
within tags/attributes?

Snippets in HTML tags

```
<body class="{{ body_class }}">  
  <p {{ p_attrs }}>  
    {{ content }}  
  </p>  
</body>
```

Option 1: Search parents until you find a sane container.

Snippets in HTML tags

```
<body class="{{ body_class }}">  
  <p {{ p_attrs }}>  
    {{ content }}  
  </p>  
</body>
```

Option 2: Parse HTML and provide selectors for diffs.

```
// data = from ajax.  
$('body').attr('class', data['body_class']);  
$('body > p:nth-child(1)').attr(data['p_attrs']);  
$('body > p:nth-child(1)').html(data['content']);
```

Hooking into the HTML: Option 1

<http://example.com/child1/>

```
<html><body><h1>
```

```
<div id="dj_block_h1">
```

```
h1 first child
```

```
</div>
```

```
</h1><div id="content">
```

```
<div id="dj_block_content">
```

```
content <p>Welcome to the first child!</p>
```

```
</div>
```

```
</div>
```


Hooking into the HTML: Option 2

<http://example.com/child1/>

```
<html><body><h1>
```

```
h1 First child
```

```
</h1><div id="content">
```

```
content <p>Welcome to the first child!</p>
```

```
</div>
```

<http://example.com/child2/>

```
{  
  "h1": "Second child",  
  "#content": "<p>Welcome to the second kid!</p>"  
}
```

Hooking into the HTML: Option 2

Instead of sending data...

```
{  
  "h1": "Second child",  
  "#content": "<p>Welcome to the second kid!</p>"  
}
```

...send instructions:

```
[  
  "$('h1').html('Second child')",  
  "$('#content').html('<p>Welcome to the second kid!</p>')"  
]
```

Evil HTML snippets

```
<div{{ rest_of_div }}  
</div>
```

rest_of_div

class="foo">

Evil HTML snippets

```
<div{{ rest_of_div }}  
</div>
```

rest_of_div

class="foo">

Solution: Don't do that.

Insertion types

Block

```
<div>{{ foo }}</div>
```


Insertion types

Block

```
<div>{{ foo }}</div>
```

Inline

```
<span>{{ foo }}</span>
```

Insertion types

Block

```
<div>{{ foo }}</div>
```

Inline

```
<span>{{ foo }}</span>
```

Text

```
<pre>{{ foo }}</pre>
```

Insertion types

Block

```
<div>{{ foo }}</div>
```

Inline

```
<span>{{ foo }}</span>
```

Text

```
<pre>{{ foo }}</pre>
```

Title

```
<title>{{ foo }}</title>
```

Insertion types

Block

```
<div>{{ foo }}</div>
```

Inline

```
<span>{{ foo }}</span>
```

Text

```
<pre>{{ foo }}</pre>
```

Title

```
<title>{{ foo }}</title>
```

Titlebit

```
<title>{{ foo }} | Example.com</title>
```

Insertion types

Block

```
<div>{{ foo }}</div>
```

Inline

```
<span>{{ foo }}</span>
```

Text

```
<pre>{{ foo }}</pre>
```

Title

```
<title>{{ foo }}</title>
```

Titlebit

```
<title>{{ foo }} | Example.com</title>
```

CSS

```
<style type="text/css">
{{ foo }}
</style>
```

Insertion types

Block

```
<div>{{ foo }}</div>
```

Inline

```
<span>{{ foo }}</span>
```

Text

```
<pre>{{ foo }}</pre>
```

Title

```
<title>{{ foo }}</title>
```

Titlebit

```
<title>{{ foo }} | Example.com</title>
```

CSS

```
<style type="text/css">
{{ foo }}
</style>
```

Dull

```
<head>
{{ foo }}<!-- meta tags -->
</head>
```


Insertion types

Block

```
<div>{{ foo }}</div>
```

Inline

```
<span>{{ foo }}</span>
```

Text

```
<pre>{{ foo }}</pre>
```

Title

```
<title>{{ foo }}</title>
```

Titlebit

```
<title>{{ foo }} | Example.com</title>
```

CSS

```
<style type="text/css">
{{ foo }}
</style>
```

Dull

```
<head>
{{ foo }}<!-- meta tags -->
</head>
```

Critical

```
<script>
window.onload=function() {
{% block onloadjs %}{% endblock %}
}
</script>
```

Template logic

```
<body>  
{% if some_condition %}  
    Some condition is true!  
{% else %}  
    It's false.  
{% endif %}  
</body>
```

Template logic

```
<body>  
{% if some_condition %}  
    Some condition is true!  
{% else %}  
    It's false.  
{% endif %}  
</body>
```

Treat the whole block as a changeable thing.

Nested template logic

```
<body>
{% if condition %}
  Condition is true!
  {% if sub_condition %}
    And sub-condition!
  {% else %}
    But not sub-condition.
  {% endif %}
{% else %}
  Condition is false.
{% endif %}
</body>
```

Treat each block as a changeable thing.

Template logic in crafty places

```
<body class="{% if foo %}foo{% else %}bar{% endif %}">
```

Removal of markup

```
<body>  
{% if condition %}  
  Condition is true!  
  {% if sub_condition %}  
    And sub-condition!  
  {% else %}  
    But not sub-condition.  
  {% endif %}  
{% else %}  
  Condition is false.  
{% endif %}  
</body>
```

condition=True
sub_condition=False

condition=True
sub_condition=True

condition=False
sub_condition=True

```
<body>  
Condition is true!  
But not sub-condition.  
</body>
```

```
<body>  
Condition is true!  
And sub-condition!  
</body>
```

```
<body>  
Condition is false.  
</body>
```


Putting it all together

Template
(as written)

```
<html>  
<head><title>{{ title }}</title></head>  
<body>  
<h1>Welcome, {{ name }}!</h1>  
</body>
```

Putting it all together

Template
(as written)

```
<html>
<head><title>{{ title }}</title></head>
<body>
<h1>Welcome, {{ name }}!</h1>
</body>
```

Rendered
HTML

```
<html>
<head><title>Title here</title></head>
<body>
<h1>Welcome, Adrian!</h1>
</body>
```

title

Title here

name

Adrian

Putting it all together

Template
(as written)

```
<html>
<head><title>{{ title }}</title></head>
<body>
<h1>Welcome, {{ name }}!</h1>
</body>
```

Rendered
HTML

```
<html>
<head><title>Title here</title></head>
<body>
<h1>Welcome, Adrian!</h1>
</body>
```

title	Title here	name	Adrian
--------------	------------	-------------	--------

Ajax request

```
GET /page2/
X-pjax: /page1/
```

Putting it all together

Template
(as written)

```
<html>
<head><title>{{ title }}</title></head>
<body>
<h1>Welcome, {{ name }}!</h1>
</body>
```

Rendered
HTML

```
<html>
<head><title>Title here</title></head>
<body>
<h1>Welcome, Adrian!</h1>
</body>
```

title Title here **name** Adrian

Ajax request

```
GET /page2/
X-pjax: /page1/
```

Ajax response

```
["document.title='Second page'",
"$('h1').html('Welcome, Bob!')"]
```

Putting it all together

Template
(as written)

```
<html>
<head><title>{{ title }}</title></head>
<body>
<h1>Welcome, {{ name }}!</h1>
</body>
```

Rendered
HTML

```
<html>
<head><title>Title here</title></head>
<body>
<h1>Welcome, Adrian!</h1>
</body>
```

title	Title here	name	Adrian
--------------	------------	-------------	--------

Ajax request

```
GET /page2/
X-pjax: /page1/
```

Ajax response

```
["document.title='Second page'",
"$('h1').html('Welcome, Bob!')"]
```

JS callback

```
document.title='Second page';
$('h1').html('Welcome, Bob!');
history.pushState(null, document.title, "/page2");
```

Security, part I

Internal variable names

```
<html>  
<body>  
  
<div id="content">  
Hello, {{ stupid_username }}  
</div>
```


Security, part 2

Auto-escaping

```
<html>  
<body>  
  
<div id="content">  
Hello, {{ name }}  
</div>
```

name	<script>alert('hacked you');</script>
-------------	---------------------------------------

Bonus!

Deferred loading

The screenshot shows the Google Analytics 'Overview: all accounts' page. The table lists several accounts, but the data columns (Visits, Avg. Time on Site, Bounce Rate, Completed Goals, and % Change) are all displaying 'Loading...'. This illustrates the concept of deferred loading where the data is not immediately available.

Accounts

Name↑	Visits	Avg. Time on Site	Bounce Rate	Completed Goals	Visits % Change	Actions
caravanguitars.com	Loading...	Loading...	Loading...	Loading...	N/A	
ebcode	Loading...	Loading...	Loading...	Loading...	N/A	
EveryBlock	Loading...	Loading...	Loading...	Loading...	N/A	
holovaty.com	Loading...	Loading...	Loading...	Loading...	N/A	
jacobian	Loading...	Loading...	Loading...	Loading...	N/A	
OLD blog.everyblock.com	Loading...	Loading...	Loading...	Loading...	N/A	
OLD chicago.everyblock.com	Loading...	Loading...	Loading...	Loading...	N/A	
OLD nyc.everyblock.com	Loading...	Loading...	Loading...	Loading...	N/A	
OLD sf.everyblock.com	Loading...	Loading...	Loading...	Loading...	N/A	
wilsonminer.com	Loading...	Loading...	Loading...	Loading...	N/A	

Find account:

Show rows: 10 1 of 1

© 2012 Google | [Analytics Home](#) | [Terms of Service](#) | [Privacy Policy](#) | [Contact us](#) | [Analytics Blog \(in English\)](#)

Waiting for www.google.com...

Deferred loading


```
<html>
<body>
<h1>{% block h1 %}{% endblock %}</h1>

<div id="content">
  {% defer_load %}
  {% block content %}{% endblock %}
  {% end_defer_load %}
</div>
```

Template scanner

- Scan all templates in the codebase.
- Figure out which combinations are good/bad.
- Use that data to inform auto-pjax.

“This sounds like a horrible amount of work.”



adrian@holovaty.com
tinyurl.com/django-pjax