



Online Education with Learnersourcing

Rob Miller

User Interface Design Group

MIT CSAIL

Joint work with Juho Kim, Sarah Weir,
Elena Glassman, Philip Guo, Carrie Cai,
Max Goldman, Phu Nguyen, Rishabh Singh, Jeremy Scott



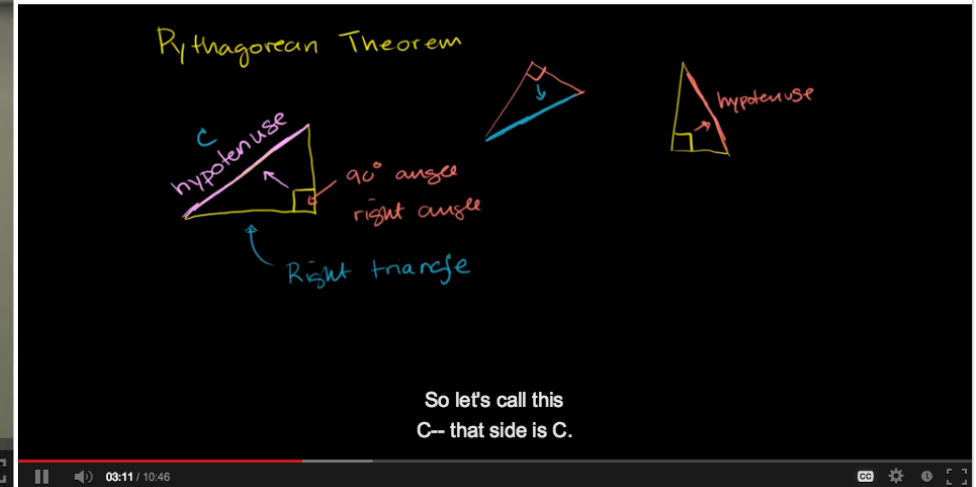
MIT | HUMAN-COMPUTER INTERACTION

Video enables learning at scale

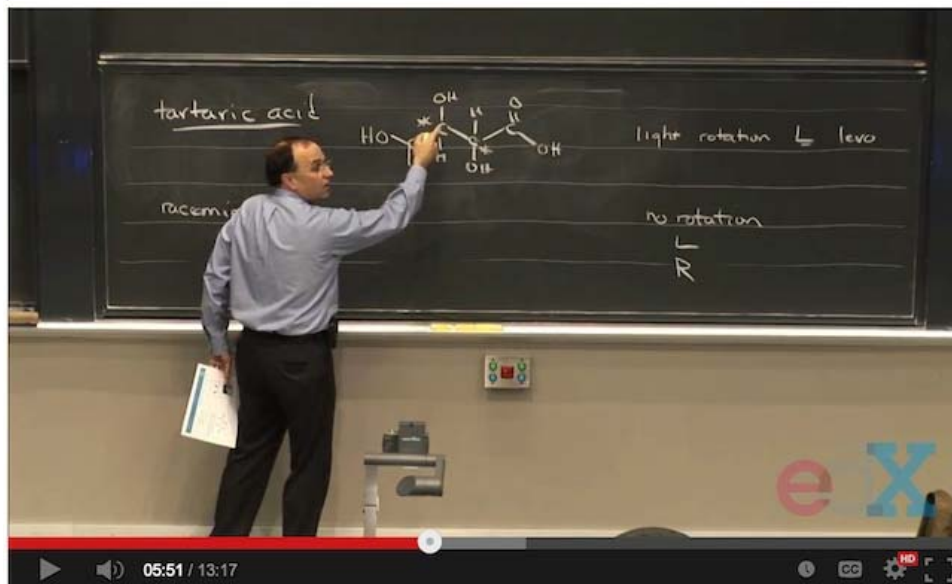


The Pythagorean Theorem
Introduction to the Pythagorean Theorem

Practice this concept



So let's call this
C-- that side is C.



Scalable delivery \neq Scalable learning



one



hundreds



millions

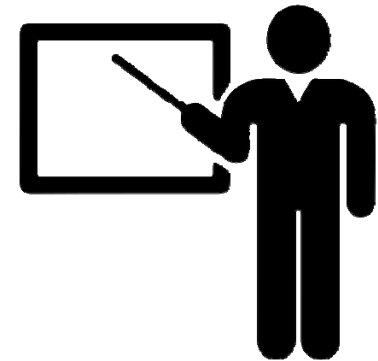
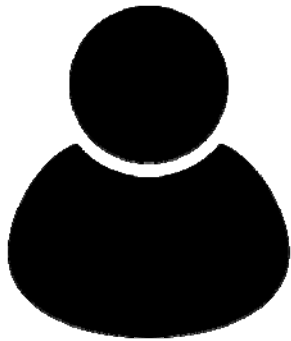
→ # of learners



In-person Learning



Direct learner-instructor interaction



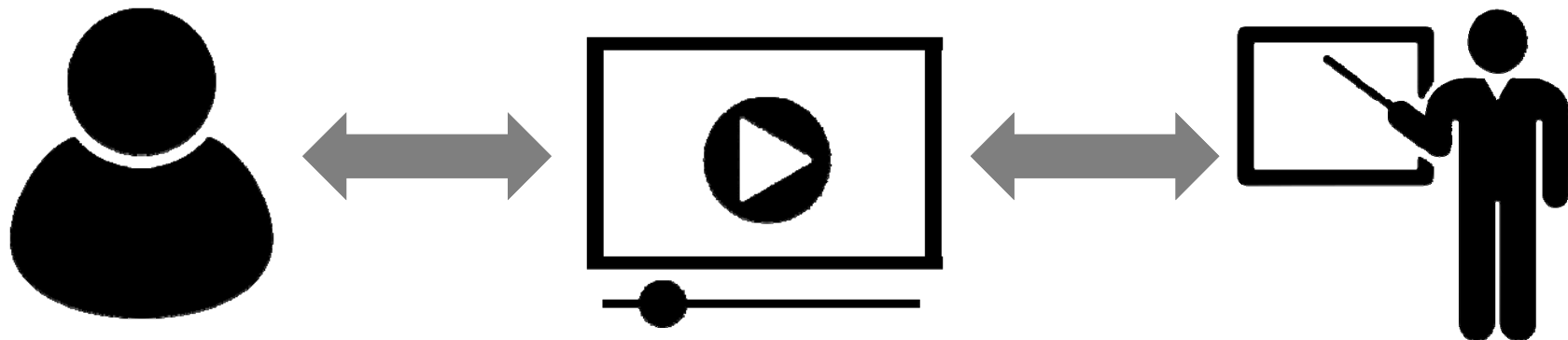
Effective pedagogy

- high engagement
- immediate feedback
- adaptive instruction



Video Learning

Mediated learner-instructor interaction



Video interfaces are limiting.

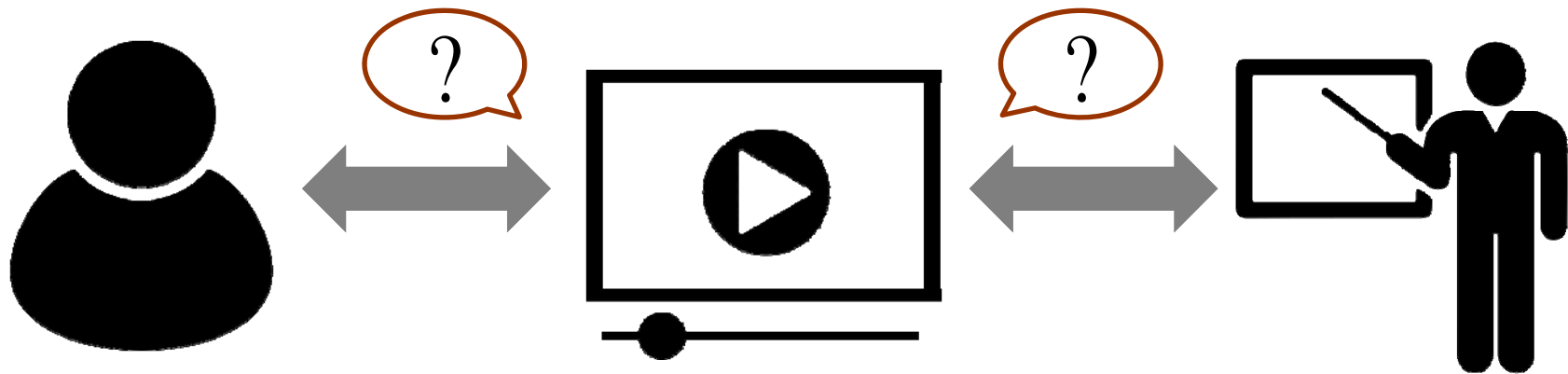
- passive, isolated viewing
- not adapting to learner
- difficult to find info and navigate freely

Problems in video learning at scale



What are the learners doing?

What is inside this video?



Crowdsourcing vs. Learnersourcing



- Crowdsourcing
 - asking a crowd to do micro-work for problems we can't solve with software
 - what do the crowds get in return? money, fun, social connection

- Learnersourcing
 - asking a crowd of **learners** to do micro-work to improve an online course
 - what do the learners get in return? **learning**
 - existing examples of learnersourcing
 - discussion forums
 - peer instruction
 - peer grading



Two Types of Learnersourcing

Passive

track what learners do



Active

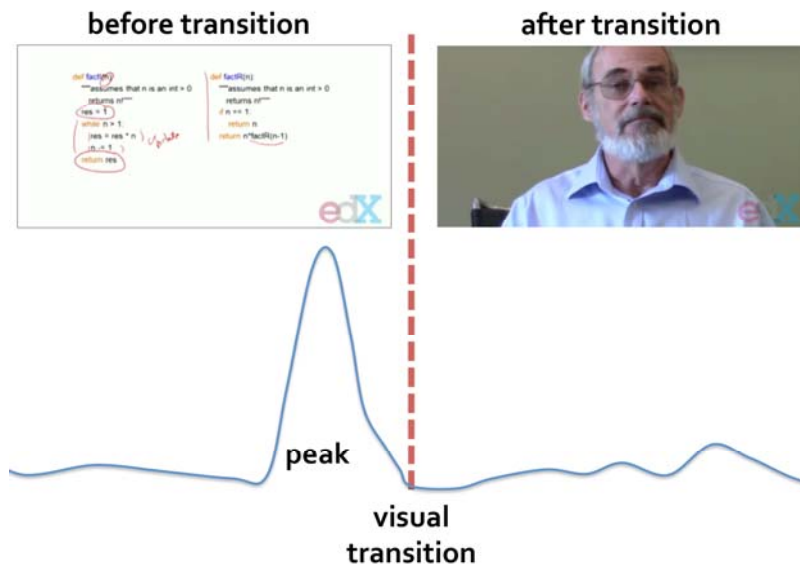
ask learners to do something



Outline of this Talk

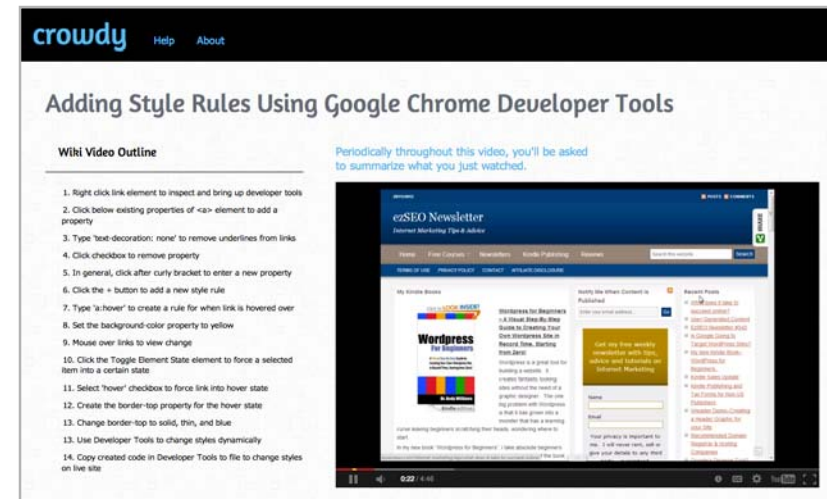
Passive

MOOC lecture videos
+ user data



Active

YouTube how-to videos
+ steps and subgoals



The screenshot shows a YouTube video player with a "Wiki Video Outline" overlay. The video title is "Adding Style Rules Using Google Chrome Developer Tools". The outline lists 14 steps for adding style rules. A "Periodically throughout this video, you'll be asked to summarize what you just watched." note is present. The video player shows a webpage with a search bar and various content blocks.

LECTURE VIDEOS

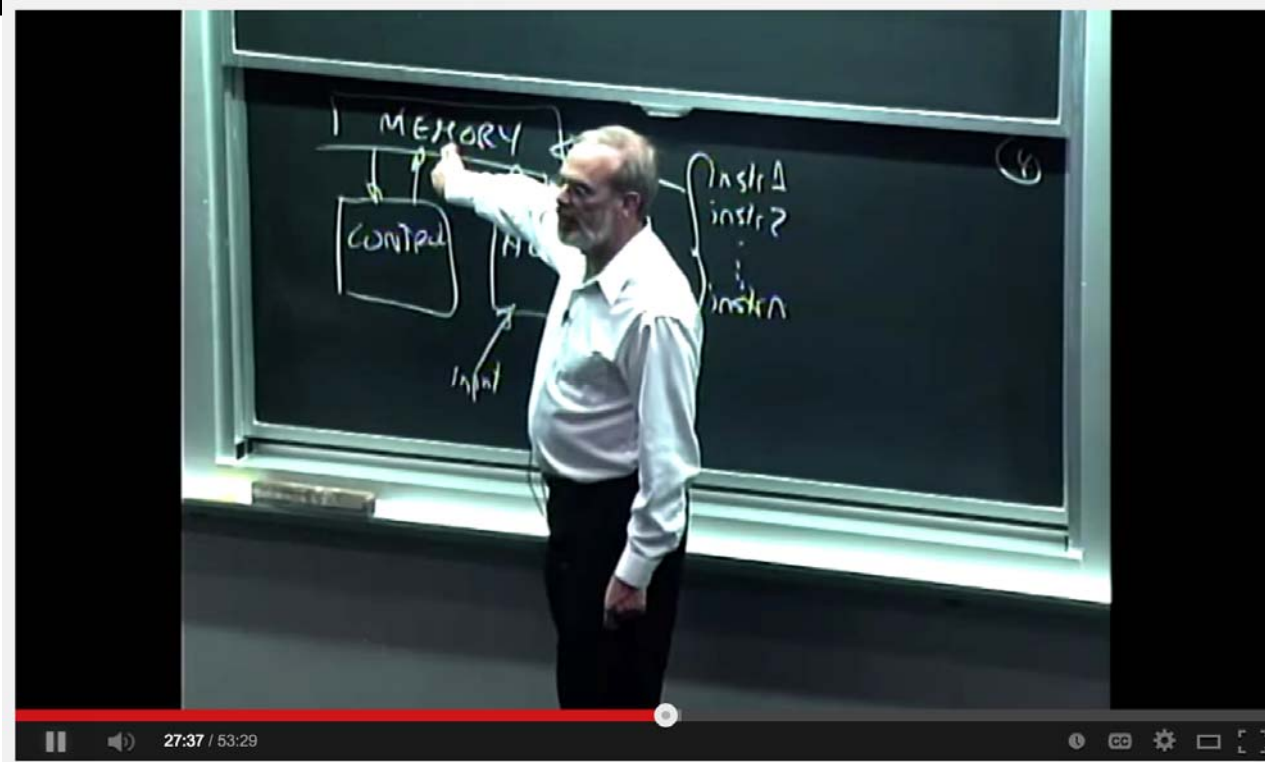


Juho Kim

MIT HUMAN-COMPUTER INTERACTION

Motivation

MOOC lecture videos are widespread, but standard video players are not optimized for learning



Challenge for instructors & editors

- We don't know how students use lecture videos
 - Confusion
 - Boredom
 - “Aha” moments
 - Re-watching important parts
- We analyzed video interaction data from the lectures in 4 edX courses
 - Clickstream (play, pause, timeline-scrub events)

Course	Subject	University	Students	Videos	Video Length	Processed Events
6.00x	Intro. CS & Programming	MIT	59,126	141	7:40	4,491,648
PH207x	Statistics for Public Health	Harvard	30,742	301	10:48	15,832,069
CS188.1x	Artificial Intelligence	Berkeley	22,690	149	4:45	14,174,203
3.091x	Solid State Chemistry	MIT	15,281	271	6:19	4,821,837
Total			127,839	862	7:46	39,319,757



How do learners navigate videos?

- Watch sequentially



- Pause



- Re-watch



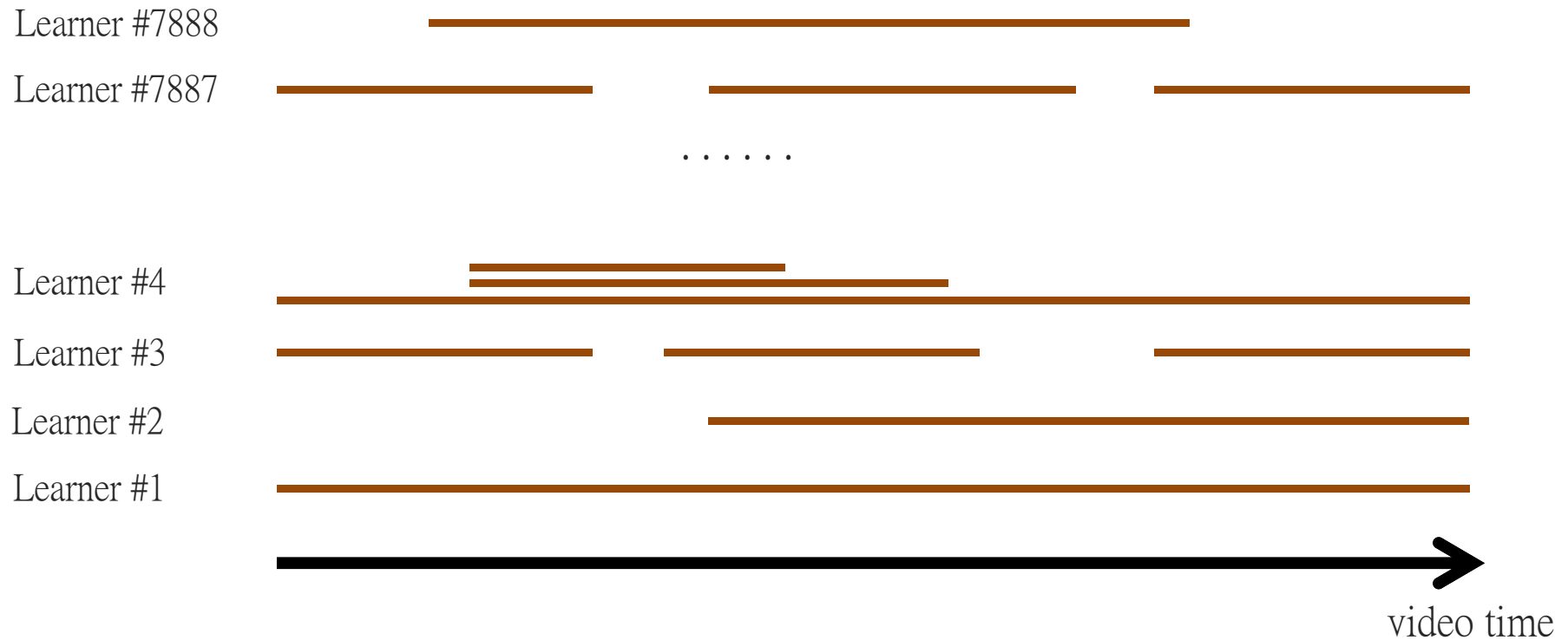
- Skip / Skim



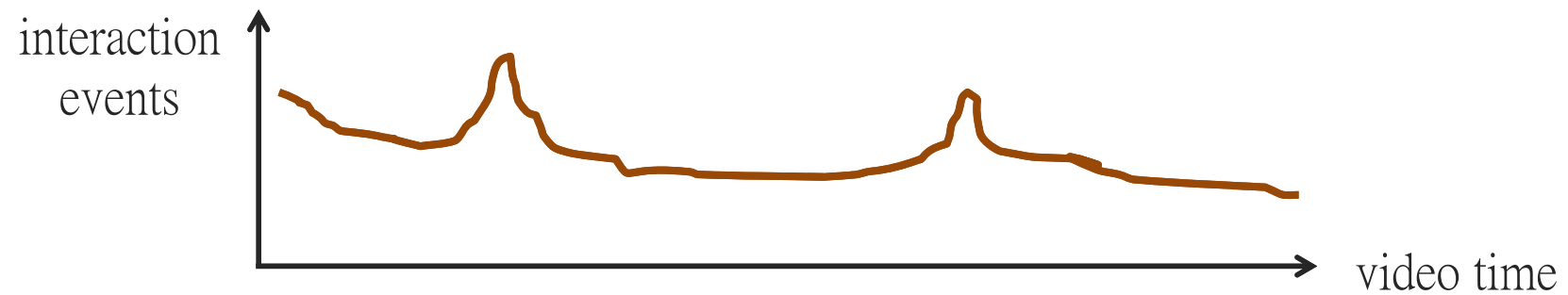
- Search



Collective Interaction Traces

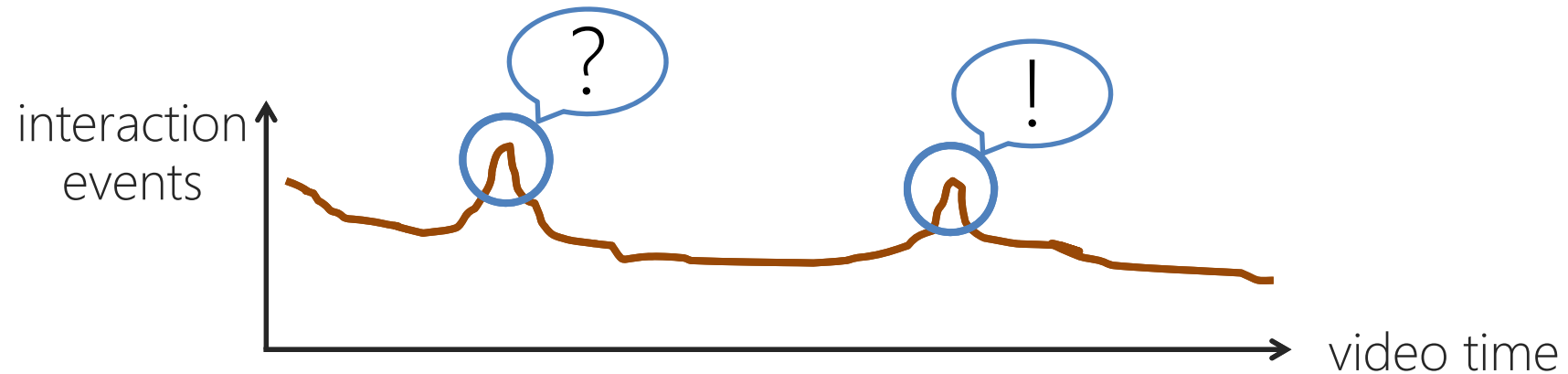


Collective Interaction Traces into Interaction Patterns

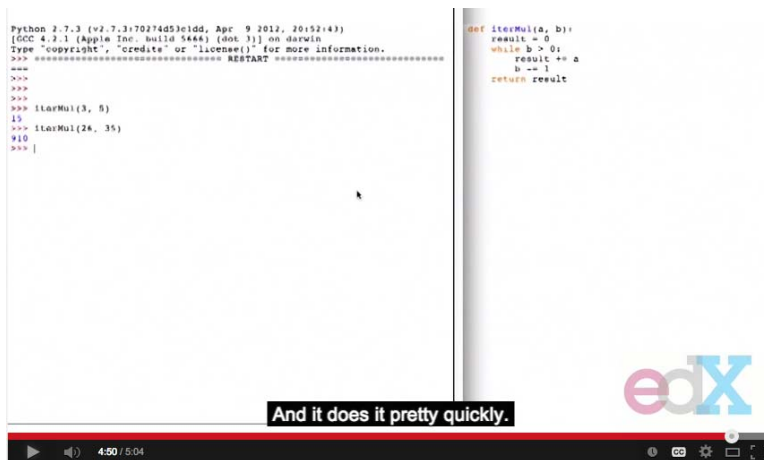


Interaction Peaks

Temporal peaks in the number of interaction events, where a significant number of learners show similar interaction patterns



Observation: **Visual / Topical transitions** in the video often coincide with a peak.




Returning to content

before transition

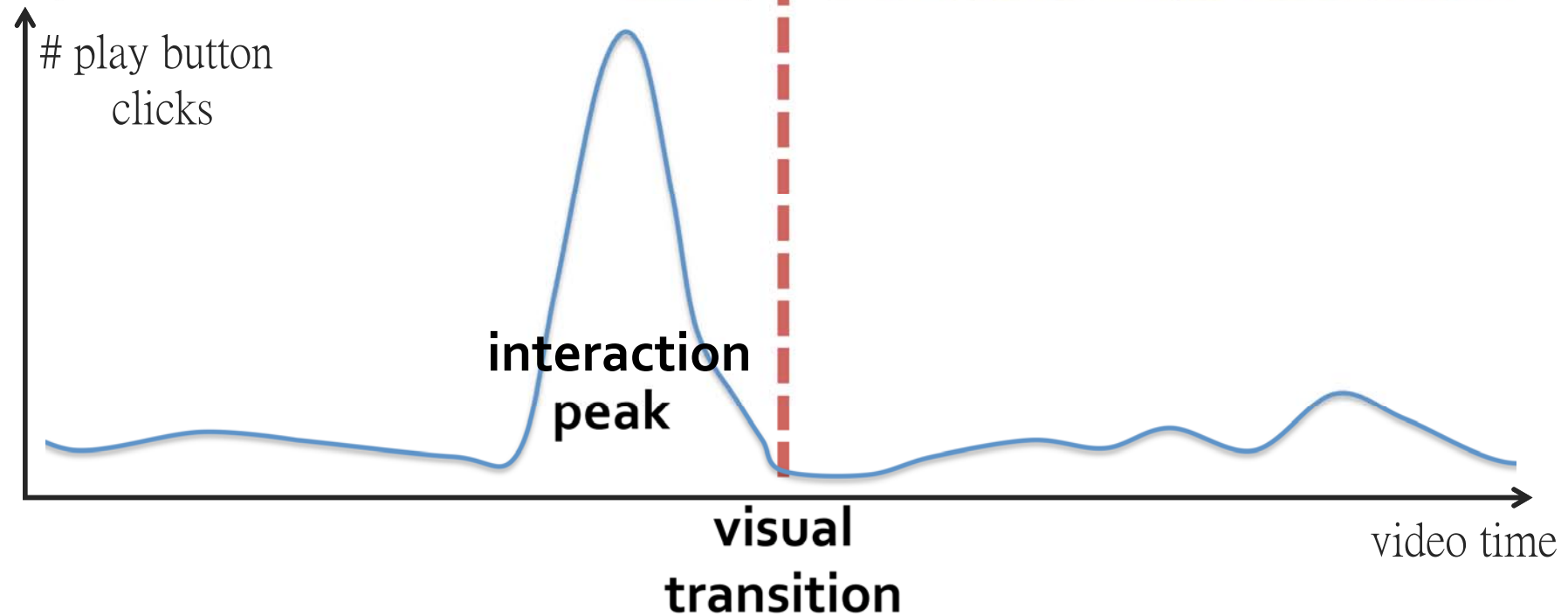
```
def fact(n):  
    """assumes that n is an int > 0  
    returns n!"""  
    res = 1  
    while n > 1:  
        res = res * n  
        n -= 1  
    return res
```

update

```
def factR(n):  
    """assumes that n is an int > 0  
    returns n!"""  
    if n == 1:  
        return n  
    return n * factR(n-1)
```




after transition



Beginning of new material

before transition

Idea: Admissibility



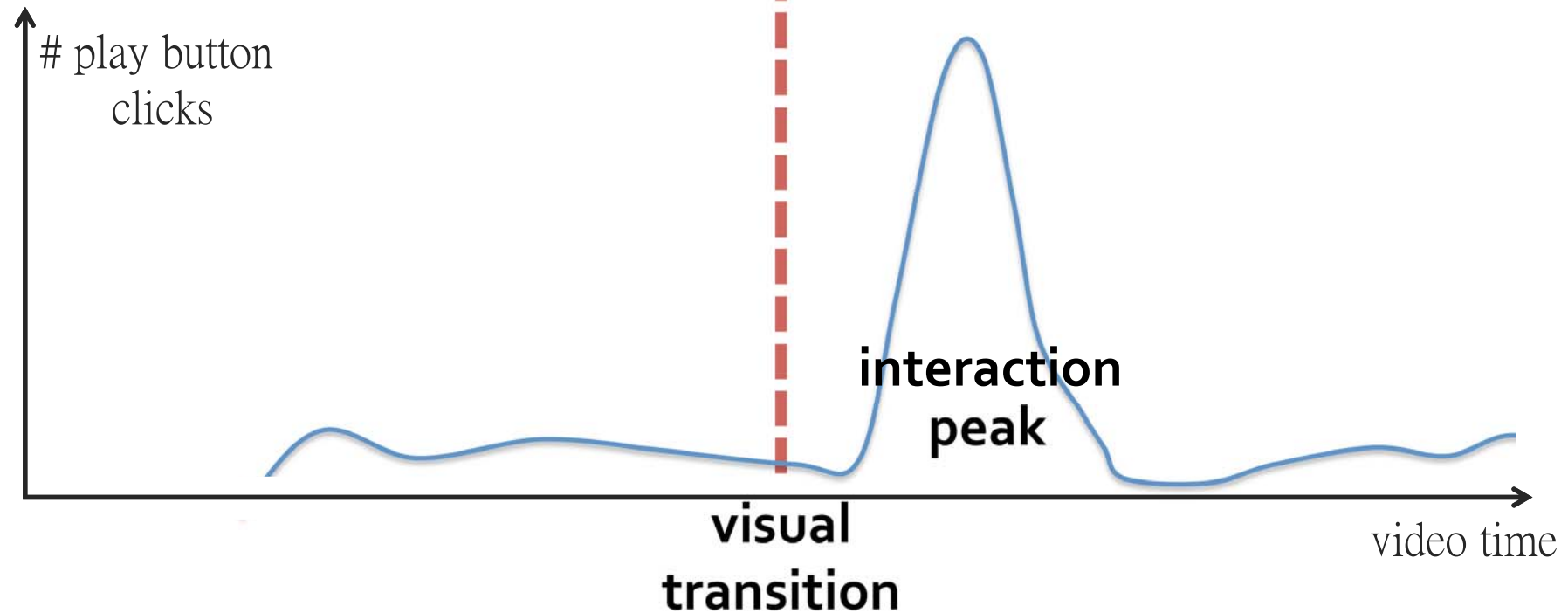
Inadmissible (pessimistic) heuristics break optimality by trapping good plans on the fringe

Admissible (optimistic) heuristics slow down bad plans but never outweigh true costs

after transition

Admissible Heuristics

- A heuristic h is *admissible* (optimistic) if:
$$0 \leq h(n) \leq h^*(n)$$
where $h^*(n)$ is the true cost to a nearest goal

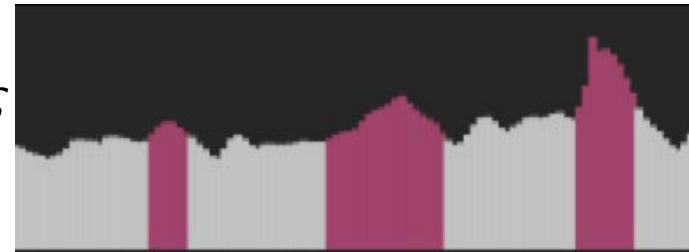


Key Features of LectureScape



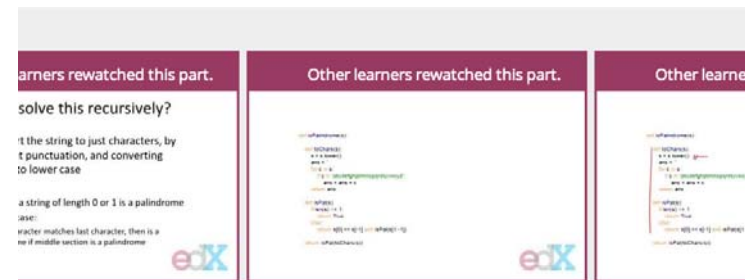
Rollercoaster Timeline

"Which parts did other learners find confusing or important?"



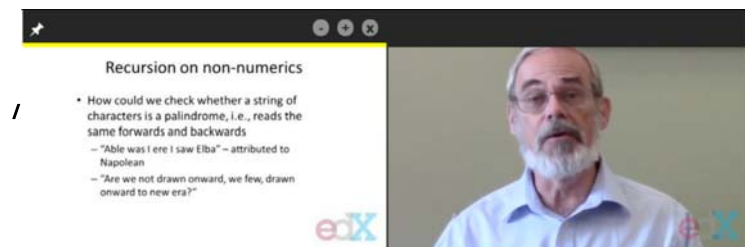
Visual Highlights

"I want a quick overview of this clip."



Automatic Pinning

"Wait, I need to see that last slide."



Lab Study



- 12 edX & On-Campus Students
- LectureScape vs baseline interface
- Navigation & learning tasks

Visual search

“Find a slide where the instructor displays on screen examples of the singleton operation.”

Problem search

“If the step size in an approximation method decreases, does the code run faster or slower?”

Summarization

“write down the main points of a video in three minutes. minutes.”



Results



With LectureScape:

- more non-linear jumps in navigation
- more navigation options
 - rollercoaster timeline
 - phantom cursor
 - highlight summary
 - pinning

*“[LectureScape] gives you more options.
It personalizes the strategy I can use in the task.”*



A sense of “learning together”



Interaction peaks matched with participants' points of
“confusion” (8/12) and “importance” (6/12)

*“It’s not like cold-watching.
It feels like watching with other students.”*

*“[interaction data] makes it seem more classroom-y,
as in you can compare yourself to how other students are
learning and what they need to repeat.”*



In Progress: edX Deployment



MITx: 6.00.1x Introduction to Computer Science and Programming Using Pyth...

mcpanic

Courseware Updates & News Calendar Wiki Discussion Progress

Overview

Help Entrance Survey

Week 1

Week 2

Week 3

Lecture 5 - Recursion
Lecture Sequence

Lecture 6 - Objects
Lecture Sequence

Problem Set 3
Problem Set due Jul 03, 2014 at
23:30 UTC

Week 4

Quiz

Week 5

Week 6

Week 7



Recursion on Strings

search for keywords inside the video

palindrome left onward string palindrome ispalindrome

How to solve this recursively?

- First, convert the string to just characters, by stripping out punctuation, and converting upper case to lower case
- Then
 - Base case: a string of length 0 or 1 is a palindrome

edX

1:44 / 7:29 Add Bookmark [4:06-4:25] You watched this segment.

So how do we solve it?
We first convert the string to just characters.
We'll look to that in a second.
And solving it recursively is actually pretty easy.
If I have a string that's either of length zero or of length one, it's a palindrome.
So length one is just one character.
Otherwise, to solve this, what I'm going to do is take the string and ask the following question.
If the first and last character are the same, then they satisfy the condition.
And let me then simply look at the remaining string, throwing away the first and last character, and ask is that a palindrome.
Wonderful.
There's that recursive property again.
If I can break it down into that problem, I'm set.
So I could write code to do that.
Just to give you the example again, this says I'm going to take something like "Able was I were I saw Elba" and reduce it to just that string of characters without the spaces or any punctuation.
And then, to test whether that string is a palindrome, that's the same as asking are the first and last characters the same?

my bookmarks frequently watched by others

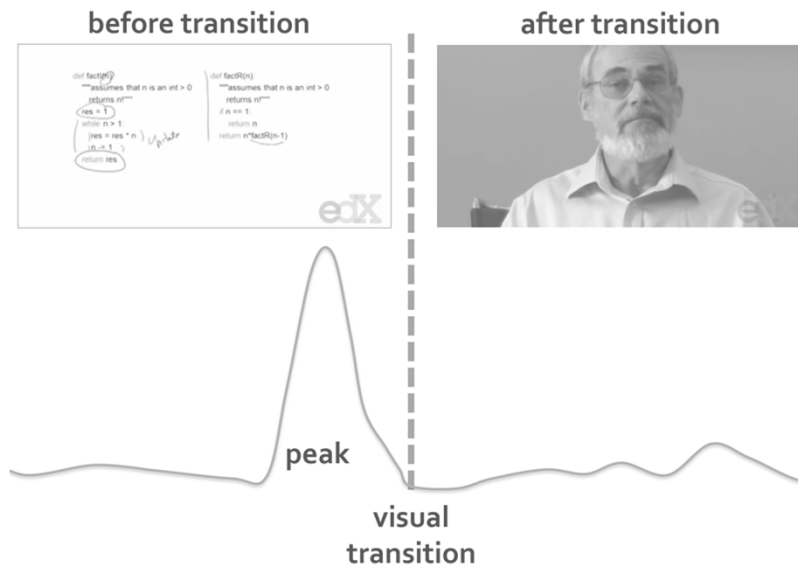
Other learners watched this part.	Other learners watched this part.	Other learners watched this part.	Other learners watched this part.
<p>Recursion on non-numerics</p> <ul style="list-style-type: none">• How could we check whether a string of characters is a palindrome, i.e., reads the same forwards and backwards<ul style="list-style-type: none">– "Able was I were I saw Elba" – attributed to Napoleon– "Are we not drawn onward, we few, drawn onward to new era?"	<p>How to solve this recursively?</p> <ul style="list-style-type: none">• First, convert the string to just characters, by stripping out punctuation, and converting upper case to lower case• Then<ul style="list-style-type: none">– Base case: a string of length 0 or 1 is a palindrome– Recursion case:<ul style="list-style-type: none">• If first character matches last character, then it is a palindrome if middle section is a palindrome	<pre>def is_palindrome(s): s = s.lower().replace(" ", "") if len(s) <= 1: return True if s[0] == s[-1]: return is_palindrome(s[1:-1]) return False</pre>	<pre>def is_palindrome(s): s = s.lower().replace(" ", "") if len(s) <= 1: return True if s[0] == s[-1]: return is_palindrome(s[1:-1]) return False</pre>



Outline of this Talk

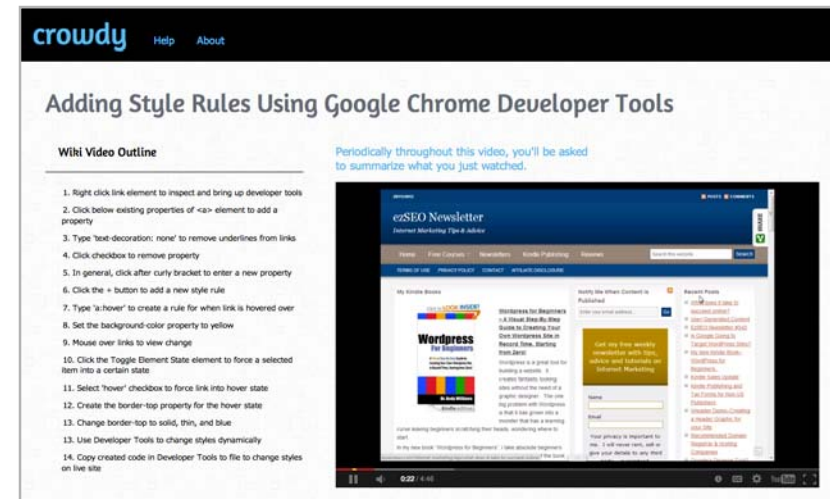
Passive

MOOC lecture videos
+ user data



Active

how-to videos
+ steps and subgoals



The screenshot shows a video player interface for a video titled 'Adding Style Rules Using Google Chrome Developer Tools' from the channel 'crowdy'. The video has a 'Wiki Video Outline' section with 14 numbered steps. A 'Periodically throughout this video, you'll be asked to summarize what you just watched.' section is also present. The video player shows a progress bar at 0:22 / 4:01.

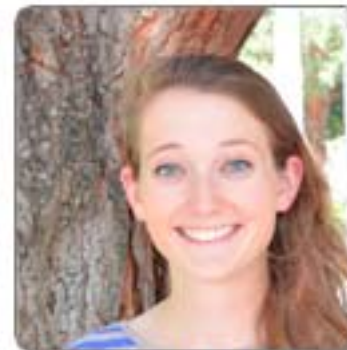
Wiki Video Outline

1. Right click link element to inspect and bring up developer tools
2. Click below existing properties of <a> element to add a property
3. Type 'text-decoration: none' to remove underlines from links
4. Click checkbox to remove property
5. In general, click after curly bracket to enter a new property
6. Click the + button to add a new style rule
7. Type 'a:hover' to create a rule for when link is hovered over
8. Set the background-color property to yellow
9. Mouse over links to view change
10. Click the Toggle Element State element to force a selected item into a certain state
11. Select 'hover' checkbox to force link into hover state
12. Create the border-top property for the hover state
13. Change border-top to solid, thin, and blue
14. Use Developer Tools to change styles dynamically

14. Copy created code in Developer Tools to file to change styles on live site



HOW-TO VIDEOS



Sarah Weir



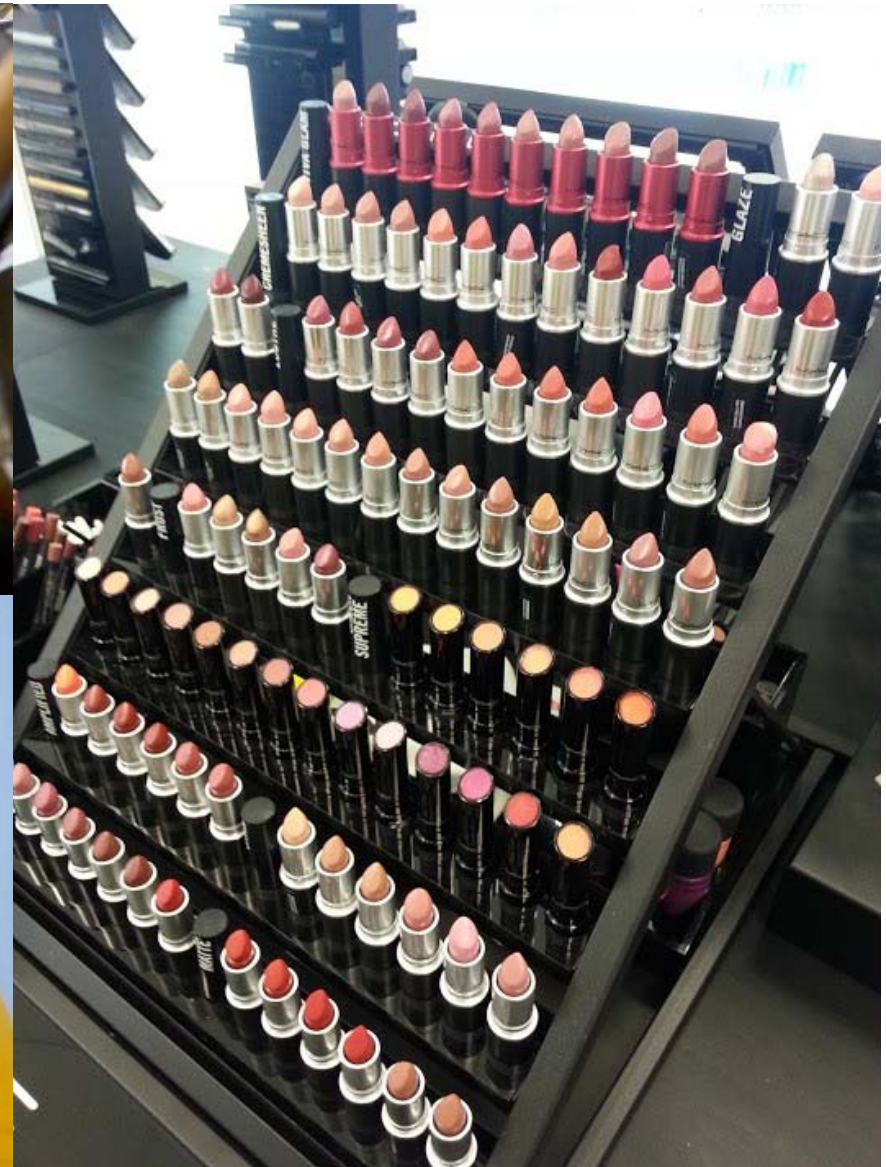
Juho Kim

How-to videos are everywhere



9

Insert the side covers with the flat end facing down.



Problems in Watching How-to Videos



- You can't tell what happens in the video.
- It's hard to go to specific parts you're interested in.



How to Bake a Cake



How-to Videos Need Steps



Overall goal

How to Make a Cake



How-to Videos Need Steps



Overall goal

How to Make a Cake

Individual steps

1. Put 2 cups of water into a mixing bowl
2. Add 1 cup of sugar to the bowl
3. Add 2 tbsp of baking soda
4. Add 1/2 tsp of salt
5. Stir together

Individual steps

6. In another bowl, beat two eggs
7. Add 1 stick of butter and beat
8. Add 1 cup of milk and stir



Subgoals Help Learning



Overall goal

How to Make a Cake

Subgoal

Combine the dry ingredients

Individual steps

1. Put 2 cups of water into a mixing bowl
2. Add 1 cup of sugar to the bowl
3. Add 2 tbsp of baking soda
4. Add 1/2 tsp of salt
5. Stir together

Subgoal

Separately combine wet ingredients

Individual steps

6. In another bowl, beat two eggs
7. Add 1 stick of butter and beat
8. Add 1 cup of milk and stir



Video Player With Steps & Subgoals



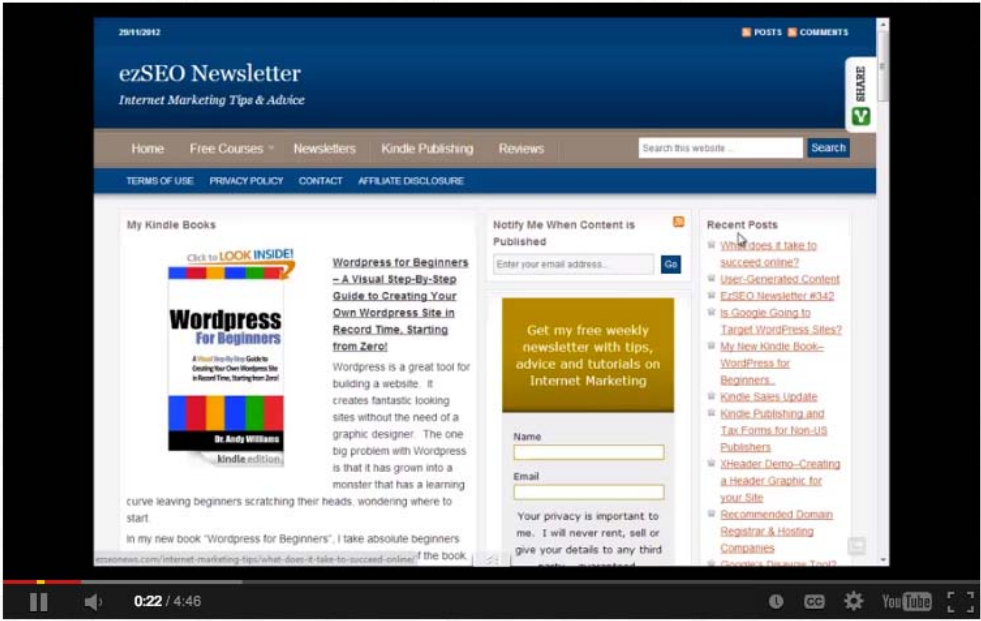
crowdy Help About

Adding Style Rules Using Google Chrome Developer Tools

Wiki Video Outline

1. Right click link element to inspect and bring up developer tools
2. Click below existing properties of <a> element to add a property
3. Type 'text-decoration: none' to remove underlines from links
4. Click checkbox to remove property
5. In general, click after curly bracket to enter a new property
6. Click the + button to add a new style rule
7. Type 'a:hover' to create a rule for when link is hovered over
8. Set the background-color property to yellow
9. Mouse over links to view change
10. Click the Toggle Element State element to force a selected item into a certain state
11. Select 'hover' checkbox to force link into hover state
12. Create the border-top property for the hover state
13. Change border-top to solid, thin, and blue
13. Use Developer Tools to change styles dynamically
14. Copy created code in Developer Tools to file to change styles on live site

Periodically throughout this video, you'll be asked to summarize what you just watched.



Video Player With Steps & Subgoals



Adding Style Rules Using Google Chrome Developer Tools

Wiki Video Outline

1. Right click link element to inspect and bring up developer tools
2. Click below existing properties of <a> element to add a property
3. Type 'text-decoration: none' to remove underlines from links
4. Click checkbox to remove property
5. In general, click after curly bracket to enter a new property
6. Click the + button to add a new style rule
7. Type 'a:hover' to create a rule for when link is hovered over
8. Set the background-color property to yellow
9. Mouse over links to view change
10. Click the Toggle Element State element to force a selected item into a certain state
11. Select 'hover' checkbox to force link into hover state
12. Create the border-top property for the hover state
13. Change border-top to solid, thin, and blue
13. Use Developer Tools to change styles dynamically
14. Copy created code in Developer Tools to file to change styles on live site

Periodically throughout this video, you'll be asked to summarize what you just watched.



How to Get Subgoals & Steps?

- Ask the learners what they've just seen (**active learnersourcing**)
 - a question pops up for every minute of video watched

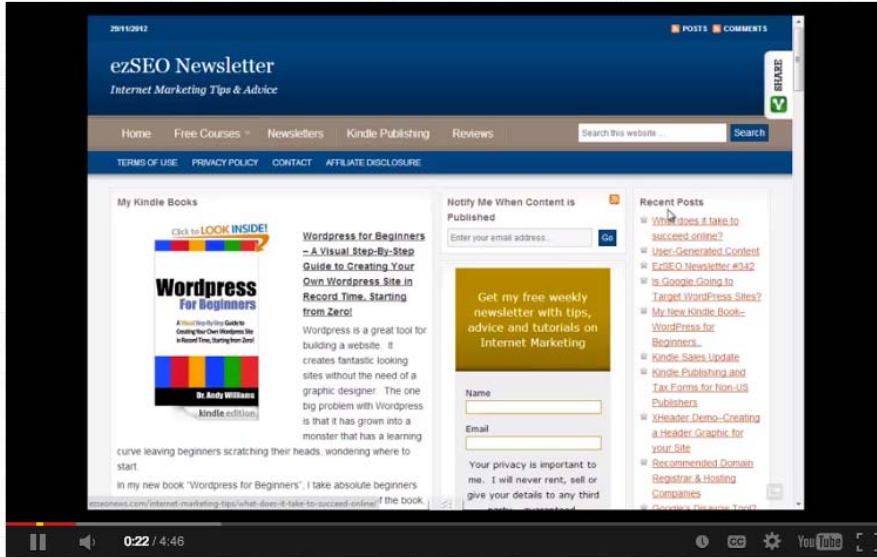
crowdy [Help](#) [About](#)

Adding Style Rules Using Google Chrome Developer Tools

Wiki Video Outline

1. Right click link element to inspect and bring up developer tools
2. Click below existing properties of <a> element to add a property
3. Type 'text-decoration: none' to remove underlines from links
4. Click checkbox to remove property
5. In general, click after curly bracket to enter a new property
6. Click the + button to add a new style rule
7. Type 'a:hover' to create a rule for when link is hovered over
8. Set the background-color property to yellow
9. Mouse over links to view change
10. Click the Toggle Element State element to force a selected item into a certain state
11. Select 'hover' checkbox to force link into hover state
12. Create the border-top property for the hover state
13. Change border-top to solid, thin, and blue
13. Use Developer Tools to change styles dynamically
14. Copy created code in Developer Tools to file to change styles on live site

Periodically throughout this video, you'll be asked to summarize what you just watched.





stage 1
generation



stage 2
evaluation



stage 3
proofreading



The screenshot shows a Crowdy.com page with a title "Adding Style Rules Using Google Chrome Developer Tools". On the left, there is a "Wiki Video Outline" with 14 numbered steps. On the right, a red-bordered box contains a question: "What was the overall goal of the video section you just watched?". Below the question is an example answer: "e.g., Create event handlers". A note states: "Note: Other users will see your outline to help them better understand the steps in the video." At the bottom of the box are a text input field, a "Submit" button, and a "Cancel" button.

crowdy Help About

Adding Style Rules Using Google Chrome Developer Tools

Wiki Video Outline

1. Right click link element to inspect and bring up developer tools
- >> 2. Click below existing properties of <a> element to add a property
3. Type 'text-decoration: none' to remove underlines from links
4. Click checkbox to remove property
5. In general, click after curly bracket to enter a new property
6. Click the + button to add a new style rule
7. Type 'a:hover' to create a rule for when link is hovered over
8. Set the background-color property to yellow
9. Mouse over links to view change
10. Click the Toggle Element State element to force a selected item into a certain state
11. Select 'hover' checkbox to force link into hover state
12. Create the border-top property for the hover state
13. Change border-top to solid, thin, and blue
13. Use Developer Tools to change styles dynamically
14. Copy created code in Developer Tools to file to change styles on live site

What was the overall goal of the video section you just watched?

e.g., Create event handlers

Note: Other users will see your outline to help them better understand the steps in the video.

 Submit Cancel

stage 1
generation



stage 2
evaluation



stage 3
proofreading



crowdy Help About

Adding Style Rules Using Google Chrome Developer Tools

Wiki Video Outline

1. Right click link element to inspect and bring up developer tools
- >> 2. Click below existing properties of <a> element to add a property
3. Type 'text-decoration: none' to remove underlines from links
4. Click checkbox to remove property
5. In general, click after curly bracket to enter a new property
6. Click the + button to add a new style rule
7. Type 'a:hover' to create a rule for when link is hovered over
8. Set the background-color property to yellow
9. Mouse over links to view change
10. Click the Toggle Element State element to force a selected item into a certain state
11. Select 'hover' checkbox to force link into hover state
12. Create the border-top property for the hover state
13. Change border-top to solid, thin, and blue
13. Use Developer Tools to change styles dynamically
14. Copy created code in Developer Tools to file to change styles on live site

Which of the following best describes the video section you just watched?

Choose the best answer (submitted by other users) or add your own.

- set up developer tools
- begin using developer tools
- developer tools
- I have a better answer:**

Submit Cancel



stage 1
generation



stage 2
evaluation



stage 3
proofreading



The screenshot shows a Crowdy interface for a proofreading task. The header includes the Crowdy logo and navigation links for 'Help' and 'About'. The main heading is 'Adding Style Rules Using Google Chrome Developer Tools'. On the left, there is a 'Wiki Video Outline' with 14 numbered steps. On the right, a red-bordered box contains a question: 'Does the below statement (submitted by other users) accurately summarize the steps?'. Below the question, the 'Statement' is 'Set up developer tools' and the 'Steps' are a single item: '1. Right click link element to inspect and bring up developer tools'. There are three radio button options: 'Yes, this statement applies', 'No, these steps don't require summarization', and 'No, and I want to revise the statement:'. A text input field contains 'Set up developer tools'. At the bottom of the box are 'Submit' and 'Cancel' buttons.




Deployment




crowdy Help About

6.813 CSS tutorials


Adding Style Rules Using Google Chrome Developer Tools
http://ezseonews.com CSS for Beginners, video 3 - Adding Style Rules Using Google Chrome Developer T
Duration: 04:47




CSS Absolute and Relative Positioning Tutorial
Follow LearnWebCode on Twitter for resources and updates: https://twitter.com/learnwebcode In this
Duration: 06:56




Making Divs Side by Side using CSS
Make two divs side by side using Float property in CSS Hey guys in this tutorial I will show you how
Duration: 04:24



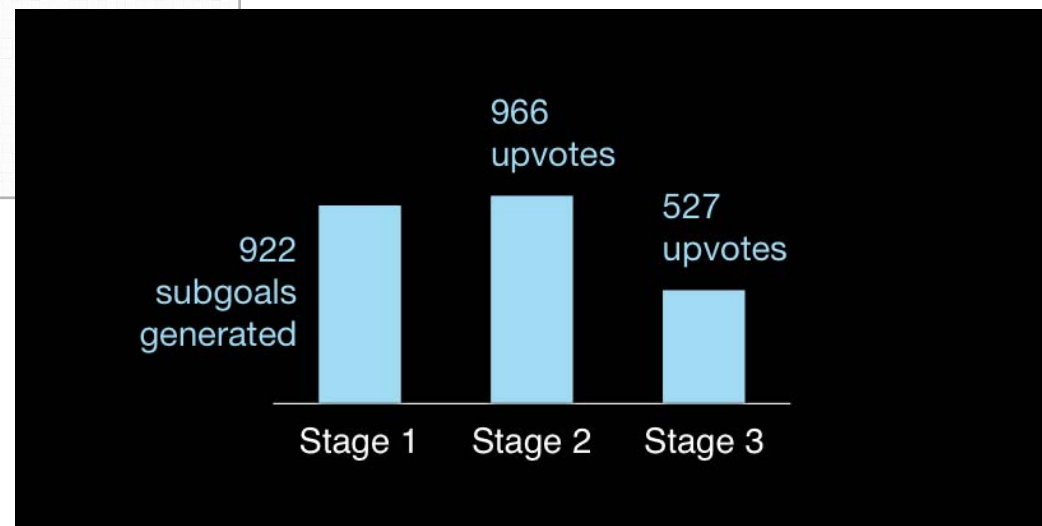
CSS (cascading style sheets) for beginners #1 - Introduction to styling with CSS
Watch more videos and ask questions at http://www.Beginnertuts.com/
Duration: 06:55



Z-Index CSS Tutorial
Follow LearnWebCode on Twitter for resources and updates: https://twitter.com/learnwebcode In this
Duration: 07:33



- 40 videos on statistics & web programming
- ~2000 visitors (~1000 participated)



Feedback from Interviews



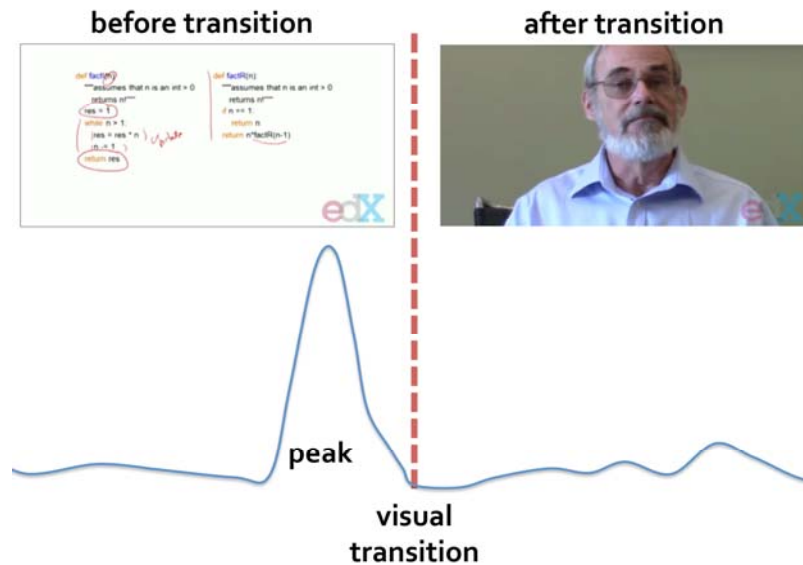
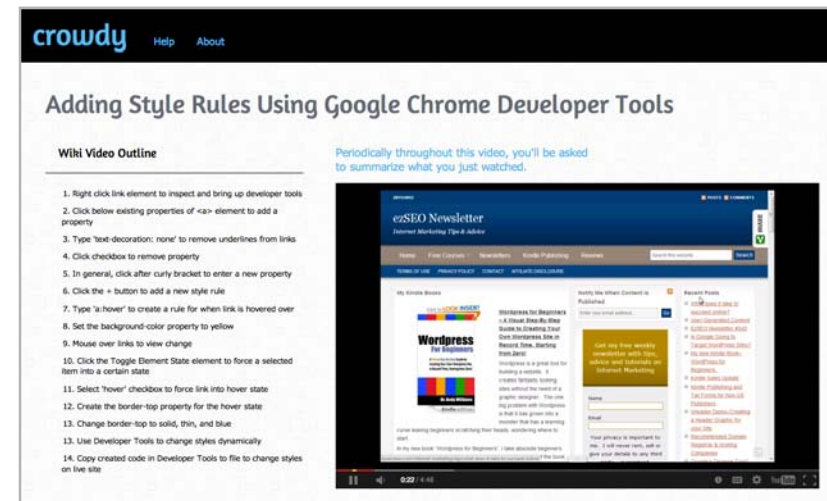
- “Having the steps there and knowing that I was going to have to fill out goals **made me pay attention** to the video slightly differently.” -- *user*
- The choices “...made me feel as though I was on the **same wavelength** still.” -- *user*
- “Having pop up questions means the viewer has to be **paying attention.**” -- *video creator*



Conclusion

Passive
 learnersourcing
 MOOC lecture videos
 + user data

Active
 learnersourcing
 how-to videos
 + steps and subgoals

crowdy Help About

Adding Style Rules Using Google Chrome Developer Tools

Wiki Video Outline

1. Right click link element to inspect and bring up developer tools
2. Click below existing properties of <a> element to add a property
3. Type 'text-decoration: none' to remove underlines from links
4. Click checkbox to remove property
5. In general, click after curly bracket to enter a new property
6. Click the + button to add a new style rule
7. Type 'a:hover' to create a rule for when link is hovered over
8. Set the background-color property to yellow
9. Mouse over links to view change
10. Click the Toggle Element State element to force a selected item into a certain state
11. Select 'hover' checkbox to force link into hover state
12. Create the border-top property for the hover state
13. Change border-top to solid, thin, and blue
14. Use Developer Tools to change styles dynamically
15. Copy created code in Developer Tools to file to change styles on live site

Periodically throughout this video, you'll be asked to summarize what you just watched.

Thanks to support from NSF, Quanta Computer, Google, edX