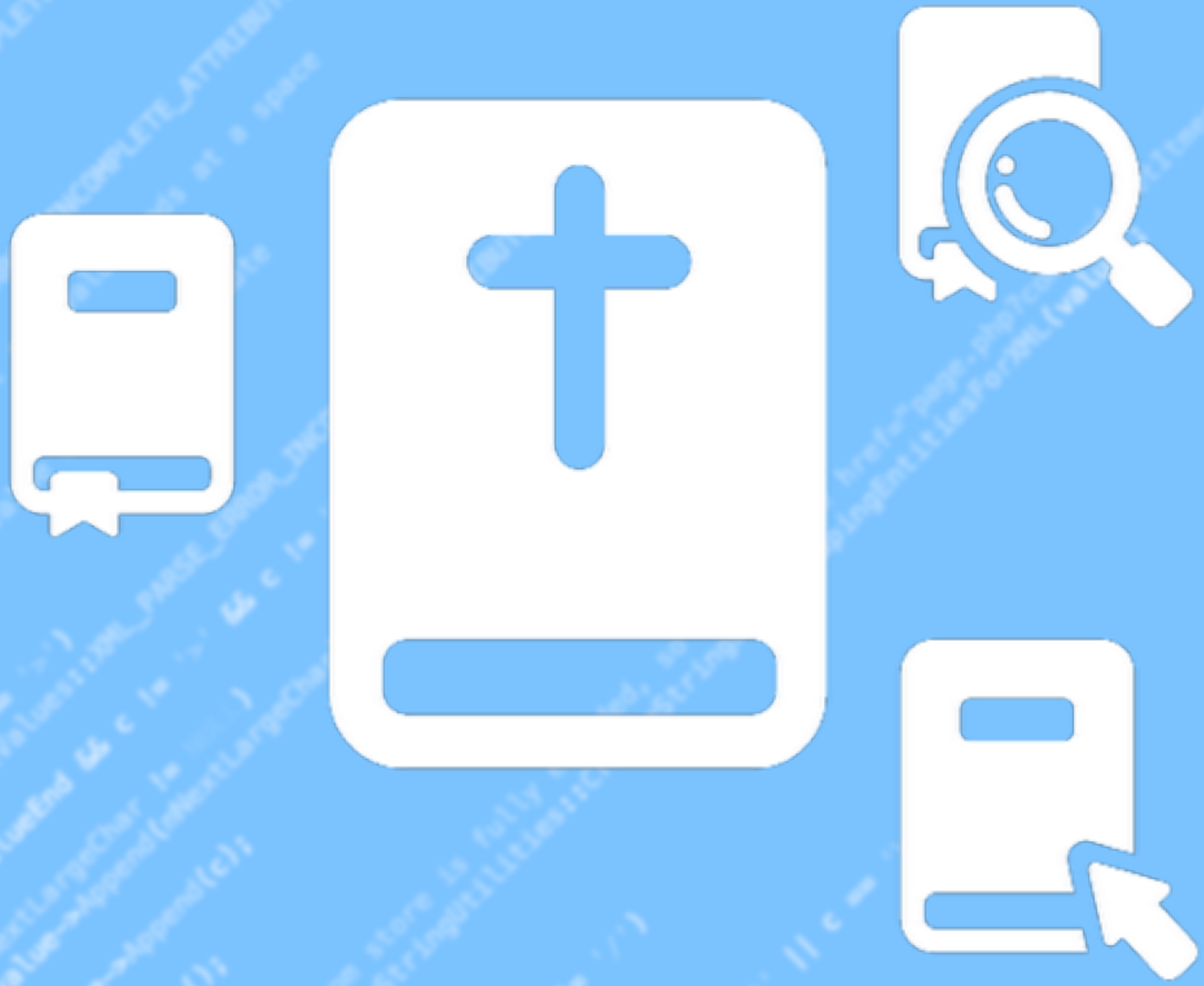
Building a better GPS for today's eText

Solving the 'where' of locations in eText

Traditional Use of Locations in Text

- Bookmarks
- Annotations
- Citations



Digital Challenges

- Resources change
- Users expect more flexibility with their annotations
- Digital text systems tend to be proprietary and not compatible with each other
- Digital locations have not been standardized

An Idea is Born

- Began work on new otPub system in Dec 2011
- Brainstormed location handling

Solution In a Nutshell

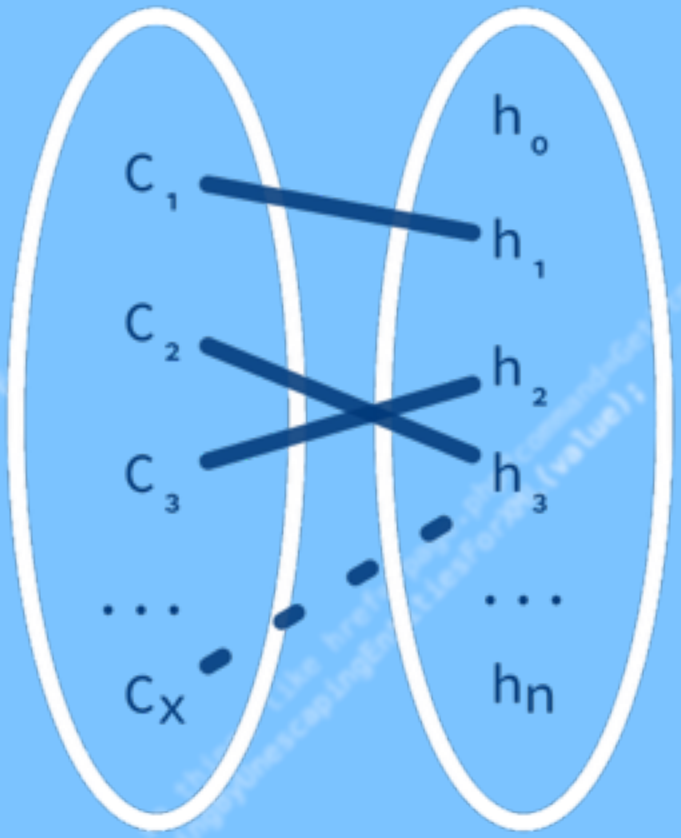
- Chunk the Text
- Hash the Chunks
- Index the Hashes
- Locations are based on these hashes and their corresponding word indexes in the text chunk

Chunk the Text

- Break the text down into chunks
- Look for logical breaking points
  - In HTML <p>,<span>,<div> are good breaking points
- Try to require 50 or more words per chunk

Hash

- Given an arbitrary length of bytes as input outputs an n-bit value known as a hash
- Many to One mapping
- A Good Hash
  - Evenly Distributed
  - Sensitive to Small Changes
- SHA256
  - Domain ~10^{75} Quattuorvigintillion
  - Low collision rate
  - In Practice '*acts*' as One to One Mapping



Hash the Chunks

- Break text based on Unicode Standard Annex #29 - a.k.a. Unicode Text Segmentation
- Ignore markup, punctuation and other non-essential elements
- Concatenate the word elements separated by a single space representing the break elements
- Collapse multiple spaces into a single space
- Hash the string using SHA256

Index the Hashes

- Build a database of hash ids to chunks and chunks to physical locations
    - Ship the database with your product
    - Build your software to refer to the index

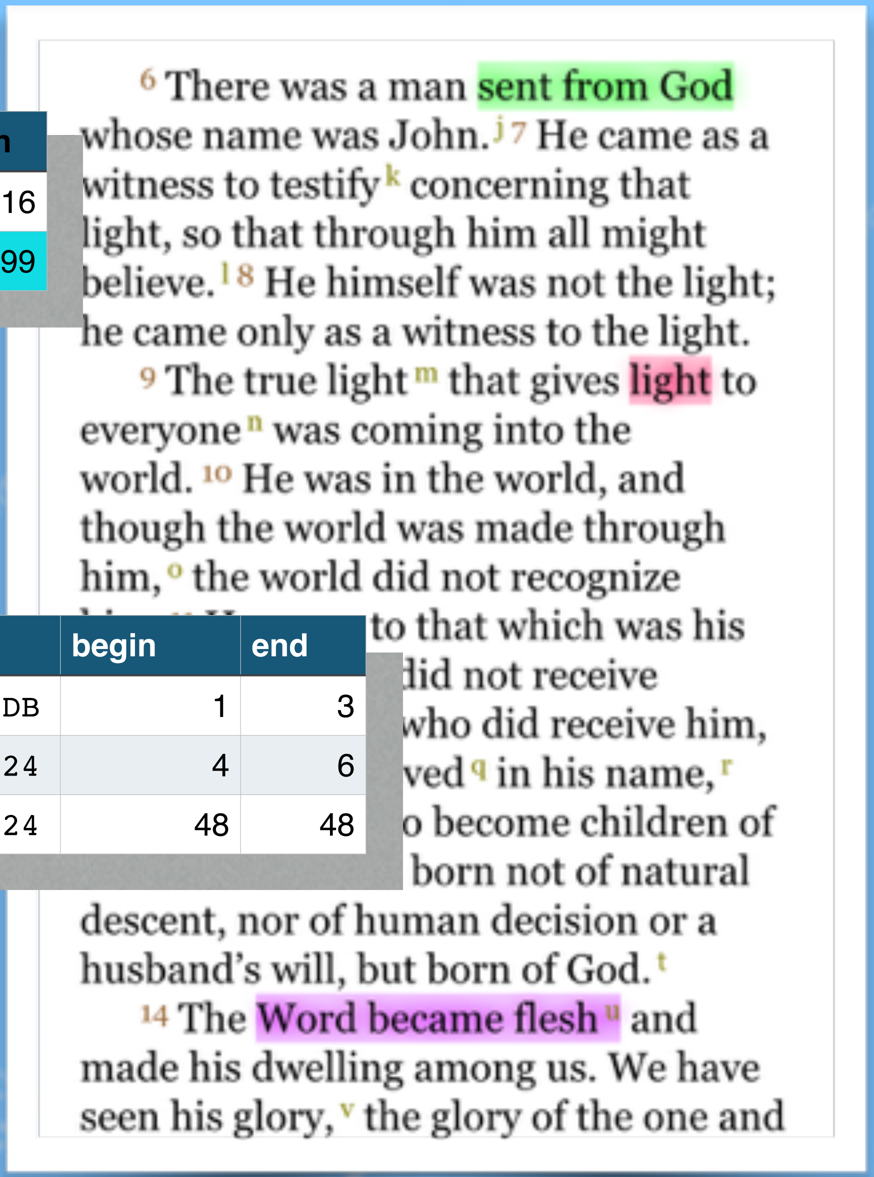
Example

Chunk Index

id	hash	file_io_location
8463	413595D0DD6D56A7BDFC0D6AC1CD9109018E7F767D6A2FF7D3632518468DB6DB	184683598616
8462	AFE0958051CEDA7A7BACF2139AE1A78FCCE3393F906E9300DB2BB6821A569324	184683596199

Annotation Locations Index

content	product	hash	begin	end
Word became flesh	17562	413595D0DD6D56A7BDFC0D6AC1CD9109018E7F767D6A2FF7D3632518468DB6DB	1	3
sent from God	17562	AFE0958051CEDA7A7BACF2139AE1A78FCCE3393F906E9300DB2BB6821A569324	4	6
light	17562	AFE0958051CEDA7A7BACF2139AE1A78FCCE3393F906E9300DB2BB6821A569324	48	48


6 There was a man sent from God
whose name was John.[j] 7 He came as a
witness to testify[k] concerning that
light, so that through him all might
believe.[l] 8 He himself was not the light;
he came only as a witness to the light.
9 The true light[m] that gives light to
everyone[n] was coming into the
world. 10 He was in the world, and
though the world was made through
him,[o] the world did not recognize
to that which was his
did not receive
who did receive him,
ved[q] in his name,[r]
o become children of
born not of natural
descent, nor of human decision or a
husband's will, but born of God.[t]
14 The Word became flesh[u] and
made his dwelling among us. We have
seen his glory,[v] the glory of the one and

- *3 Annotations*
  - *Requiring 3 Locations*
    - *Requiring 2 Chunks*

Challenges

- An indexed database will bloat the size of the resource.
- Hashing is slow
  - Only needed for resource building
- Sensitive to Chunk and Work break algorithm changes

Advantages

- Most (99+%) Hashes survive minor resource updates
- Focuses on the text content not the format
- Some things just get easier
- Olive Tree is currently using this method and we can enthusiastically say
  - **‘This Works!’**

What's Next

- Can this system be standardized?
  - Allow for sharing annotations across systems
  - Replace page number based citations?

Conclusion

- Chunk the Text
- Hash the Chunks
- Index the Hashes

Thank You

David Trotz
davidtrotz@olivetree.com

Demo:
http://olivetreebible.github.io/bt15demo

Some Icons in this presentation were made by freepik.com available from www.flaticon.com